卡尔曼滤波原理及应用
——MATLAB 仿真

黄小平　王　岩　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书主要介绍数字信号处理中的卡尔曼（Kalman）滤波算法及在相关领域应用的相关内容。全书共 7 章。第 1 章为绪论。第 2 章介绍 MATLAB 算法仿真的编程基础。第 3 章介绍线性 Kalman 滤波。第 4 章讨论扩展 Kalman 滤波，并介绍其在目标跟踪和制导领域的应用和算法仿真。第 5 章介绍 UKF 滤波算法，同时也给出其应用领域内的算法仿真实例。第 6 章介绍了交互多模型 Kalman 滤波算法。第 7 章介绍 Simulink 环境下，如何通过模块库和 S 函数构建 Kalman 滤波器，并给出了系统是线性和非线性两种情况的滤波器设计方法。

本书可以作为电子信息类各专业高年级本科生和硕士、博士研究生数字信号处理课程或者 Kalman 滤波原理的教材，也可以作为从事雷达、语音、图像等传感器数字信号处理的教师和科研人员的参考书。

图书在版编目（CIP）数据

卡尔曼滤波原理及应用：MATLAB 仿真 / 黄小平，王岩编著. —北京：电子工业出版社，2015.7
ISBN 978-7-121-26310-1

Ⅰ. ①卡… Ⅱ. ①黄… ②王… Ⅲ. ①卡尔曼滤波－系统仿真－Matlab 软件 Ⅳ. ①O211.64-39

中国版本图书馆 CIP 数据核字（2015）第 127904 号

责任编辑：刘海艳
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×1 092　1/16　印张：11.75　字数：264 千字
版　　次：2015 年 7 月第 1 版
印　　次：2015 年 7 月第 1 次印刷
印　　数：3 000 册　　定价：39.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

随着科技的发展，在雷达、声呐、通信、视频图像处理、故障诊断等领域，对信号检测和状态参数估计的研究，有着重要的价值。在所有数字信号处理应用系统中，传感器数据采集是重要的一环。所有传感器测量的数据都受到噪声的污染，噪声不能消除，只能尽最大限度地降低。例如，在目标跟踪中，传感器一般是测量观测站与目标之间的距离、角度等信息，这些信息往往会受到高斯、非高斯噪声的污染，导致观测站不能准确地估计目标的状态，那么这时对测量数据进行滤波就显得很有必要了。

卡尔曼（Kalman）滤波是噪声处理的利器，目前关于卡尔曼滤波的论文非常多，专著也不少，但是在阐述 Kalman 滤波原理时，大多文献只停留在公式推导和文字介绍上，而且各作者公式表示习惯不一样导致要理解 Kalman 滤波原理非常困难，在编程仿真上也存在诸多疑问，因此很多读者在刚开始接触该滤波算法时总是疑虑重重。鉴于此，本书在介绍 Kalman 滤波原理时，加入了大量的应用仿真实例。本书写作时尽量避免繁缛的公式推导，用通俗易懂的语言文字，配用详细的 MATLAB 仿真程序和中文注释，读者可以对照核心公式和程序注释理解卡尔曼滤波原理。

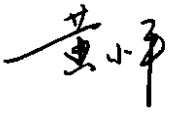
本书的主要内容是卡尔曼滤波的状态估计方法：应用在线性领域中，主要是经典卡尔曼滤波；而应用在非线性系统中，主要是扩展卡尔曼滤波和无迹 Kalman 滤波。当然在很多文献中有各种卡尔曼滤波的衍生算法，例如信息卡尔曼、强跟踪卡尔曼、集合卡尔曼、容积卡尔曼和神经网络卡尔曼等。笔者认为，其他衍生算法都是以经典卡尔曼为母体的，只要掌握经典 Kalman 滤波算法的核心和精髓即能触类旁通，学一知百。同样地，在研究各种衍生算法之前，必须先掌握经典算法。

在应用实例方面，读者一定要掌握系统建模问题。所谓系统建模，是指 Kalman 滤波中的状态方程和观测方程的建立。这两个方程中的状态、矩阵参数的设置不同，就代表着不同的系统。经典卡尔曼滤波和交互多模型卡尔曼滤波属于线性滤波器，这些算法的应用领域主要有温度测量、GPS 导航、石油地震勘探、视频图像中的目标检测和跟踪。非线性滤波器主要有 EKF 和 UKF 算法，应用实例主要是

纯方位、纯距离的目标跟踪、寻的制导系统。在工程应用中，系统模型是千奇百怪的，本书不可能列举所有的应用。鉴于此本书给出了通用的一维、二维和四维状态系统滤波问题，读者掌握这些通用模型仿真，在遇到其他信号处理模型时即可得心应手了。

参加本书编写的还有王岩、聂金平、闫芬菲、陈冰洁、田龙飞、李超、王夏静、钱琛、杨刚、李超（小）、许蓓蓓。本书的编辑和勘误，得到了北航同课题组实验室的学长的帮助，感谢王驭风、刘涛、徐建伟的指导。另外特别感谢北京理工大学何绍敏的全力相助，感谢一直支持和帮我修改错误的各位网友！

希望本书对于从事相关领域的研究者有所帮助。由于作者的水平有限，其中难免有疏漏和不足之处，恳请读者提出宝贵的意见，我的邮箱 xiaoping_444@126.com。



2015 年 4 月

目　　录

第 1 章　绪　　论

1.1　滤波的基础知识

什么是滤波？滤波一词起源于通信理论，它是从含有干扰的接收信号中提取有用信号的一种技术。而更广泛地，滤波是指利用一定的手段抑制无用信号，增强有用信号的数字信号处理过程。

无用信号，也叫噪声，是指观测数据对系统没有贡献或者起干扰作用的信号。在通信中，无用信号表现为特定波段频率、杂波；在传感器数据测量中，无用信号表现为幅度干扰。例如，在温度测量中，传感器测量值与真实温度之间往往有一定的随机波动，这个波动就是随机干扰。其实噪声是一个随机过程，而随机过程有其功率谱密度函数，功率谱密度函数的形状决定了噪声的“颜色”。如果这些干扰信号幅度分布服从高斯分布，而它的功率谱密度又是均匀分布的，则称它为高斯白噪声。高斯白噪声是大多数传感器所具有的一种测量噪声。

在工程应用中，如雷达测距、声呐测距、图像采集、声音录制等，只要是传感器采集和测量的数据，都携带噪声干扰。这种影响有的很微小，有的则会使信号变形、失真，有的严重导致数据不可用。那么滤波也不是万能的，滤波只能最大限度降低噪声的干扰，即有的滤波是不能完全消除噪声，有的则可能完全消除。

1.2　Kalman 滤波的背景

滤波理论就是在对系统可观测信号进行测量的基础上，根据一定的滤波准则，采用某种统计量最优方法，对系统的状态进行估计的理论和方法。所谓最优滤波或最优估计是指在最小方差意义下的最优滤波或估计，即要求信号或状态的最优估值应与相应的真实值的误差的方差最小。经典最优滤波理论包括 Wiener（维纳）滤波理论和 Kalman（卡尔曼）滤波理论。前者采用频域方法，后者采用时域状态空间方法。

经典 Wiener 滤波方法是由控制论创始人 N. Wiener 在 20 世纪 40 年代初（第

二次世界大战期间）由于研究火炮控制系统的需要而提出的。它是一种频域滤波方法，它的基本工具是平稳随机过程谱分解。其缺点和局限性是要求信号为平稳随机过程，要求存储全部历史数据。滤波器是非递推的，计算量和存储量大，难以在工程上实现，不便于实时应用，它仅适用于单通道平稳随机信号。人们试图将 Wiener 滤波推广到非平稳和多维的情况，都因无法突破计算上的困难而难以推广和应用。

采用频域设计法是造成 Wiener 滤波器设计困难的根本原因。因此人们逐渐转向寻求在时域内直接设计最优滤波器的方法。Kalman 在 20 世纪 60 年代初提出了 Kalman 滤波方法。Kalman 滤波方法是一种时域方法。它把状态空间的概念引入随机估计理论中，把信号过程视为白噪声作用下的一个线性系统的输出，用状态方程来描述这种输入-输出关系，估计过程中利用系统状态方程、观测方程和白噪声激励，即系统过程噪声和观测噪声，它们的统计特性形成滤波算法。由于所用的信息都是时域内的量，所以 Kalman 滤波不但可以对平稳的一维随机过程进行估计，也可以对非平稳的、多维随机过程进行估计。同时 Kalman 滤波算法是递推的，便于在计算机上实现实时应用，克服了经典 Wiener 滤波方法的缺点和局限性。

在实际应用中，Kalman 滤波是统计估计理论的里程碑式的进展，同时也是 20 世纪最伟大的发现之一。它成为众多电子系统体系中与“硅”一样不可或缺的元素。它最直接的应用是在复杂动态系统，例如连续制导过程、飞机、船舶、宇宙飞船等的控制问题上。为了实现对动态系统的控制，首先需要了解被控对象的实时状态。对于复杂动态系统应用，通常无法测量每一个需要控制的变量，而 Kalman 滤波能够利用这些有限的、不直接的、包含噪声的测量信息去估计那些缺失的信息。此外，Kalman 滤波也被用于预测动态系统未来的变化趋势，例如洪流流量、星体运动轨迹、商品交换价格等。

1.3 Kalman 滤波的发展过程

我们知道，估计的准则不同，会导致不同的估计方法。同样，利用观测序列和观测信号的方式不同，也会导致不同的估计方法。由于这两个方面的原因，滤波估计经历了最小二乘法，Wiener 滤波和 Kalman 滤波的发展而不断地完善。

最早的估计方法是德国著名数学家、物理学家、天文学家、大地测量学家——约翰·弗里德里希·高斯（C.F.Guass，1777—1855 年）于 1795 年在《天

体运动理论》一书中提出的最小二乘法。最小二乘法没有考虑被估参数和观测数据的统计特性，因此这种方法不是最优估计。由于最小二乘法在计算上比较简单，使得它成为一种应用最广泛的估计方法。1912 年英国统计与遗传学家罗纳德·费希尔（Ronald Fisher，1890—1962 年）提出了极大似然估计方法，从概率密度出发来考虑估计问题，对估计理论做出了重大贡献。

对于随机过程的估计，到 20 世纪 30 年代才积极发展起来。1940 年，控制论的创始人之一——美国学者诺伯特·维纳（Norbert Wiener，1894—1964 年）根据火力控制上的需要提出一种在频域中设计统计最优滤波器的方法，该方法被称为 Wiener 滤波。同一时期，苏联杰出数学家——安德列·柯尔莫哥洛夫（Andrey Nikolaevich Kolmogorov，1903—1987 年）提出并初次解决广义离散平稳随机序列的预测和外推问题。Wiener 滤波和柯尔莫哥洛夫滤波方法开创了一个应用统计估计方法研究随机控制问题的新领域。由于 Wiener 滤波采用频域设计法，运算复杂，解析求解困难，整批数据处理要求存储空间大，造成其适用范围极其有限，仅适用于一维平稳随机过程的信号滤波。

Wiener 滤波的缺陷促使人们寻求在时域内直接设计最优滤波器的新方法，其中匈牙利裔美国数学家——鲁道夫·卡尔曼（Rudolf Emil Kalman，1930—）的研究最具有代表性。最早实现 Kalman 滤波器的是斯坦利·施密特（Stanley Schmidt）。卡尔曼在 NASA 埃姆斯研究中心访问时发现斯密特的方法对于解决阿波罗计划的轨道预测很有用，后来阿波罗飞船的导航计算机使用了这种滤波器。1960 年，卡尔曼提出了离散系统 Kalman 滤波；1961 年，他又与布西（R.S.Bucy）合作，把这一滤波方法推广到连续时间系统中去，从而形成 Kalman 滤波设计理论。这种滤波方法采用与 Wiener 滤波相同的估计准则，二者的基本原理是一致的。但是，Kalman 滤波是一种时域滤波方法，采用状态空间方法描述系统，算法采用递推形式，数据存储量小，不仅可以处理平稳随机过程，也可以处理多维和非平稳随机过程。

正是由于 Kalman 滤波具有以上其他滤波方法所不具备的优点，Kalman 滤波理论一提出立即应用到工程实际当中。例如，阿波罗登月计划和 C-5A 飞机导航系统，就是 Kalman 滤波早期应用中最成功的实例。随着电子计算机的迅速发展和广泛应用，Kalman 滤波在工程实践中特别是在航天空间技术中迅速得到应用。目前 Kalman 滤波理论作为一种最重要的最优估计理论被广泛应用于各领域，如惯性导航、制导系统、全球定位系统、目标跟踪系统、通信与信号处理、金融等。

Kalman 最初提出的滤波基本理论只适用于线性系统，并且要求观测方程也必须是线性的。在此后的多年间，Bucy 等人致力研究 Kalman 滤波理论在非线性系统和非线性观测下的扩展 Kalman 滤波，扩展了 Kalman 滤波的适用范围。扩展 Kalman 滤波（Extended Kalman Filter，EKF）是一种应用广泛的非线性系统滤波方法。这种滤波器的思想是将非线性系统一阶线性化，然后利用标准 Kalman 滤波，其存在的问题是线性化过程会带来近似误差。

1999 年，S.Julier 提出无迹 Kalman 滤波（Unscented Kalman Filter，UKF），中文释义是无损 Kalman 滤波、无迹 Kalman 滤波或去芳香 Kalman 滤波。它是以 UT 变换为基础，采用 Kalman 线性滤波的框架，摒弃了对非线性函数进行线性化的传统做法。对于一步预测方程，使用 UT 变换来处理均值和协方差的非线性传递，就成为 UKF 算法。UKF 无需像 EKF 那样要计算 Jacobian 矩阵，无需忽略高阶项，因而计算精度较高。

Kalman 滤波应用范围广泛，设计方法也简单易行，但它必须在计算机上执行。随着微型计算机的普及应用，人们对 Kalman 滤波的数值稳定性、计算效率、实用性和有效性的要求越来越高。由于计算机的字长有限，使计算中舍入误差和截断误差累积、传递，造成误差方差阵失去对称正定性，造成数值不稳定。在 Kalman 滤波理论的发展过程中，为改善 Kalman 滤波算法的数值稳定性，并提高计算效率，人们提出平方根滤波、UD 分解滤波等一系列数值鲁棒的滤波算法。

传统的 Kalman 滤波是建立在模型精确和随机干扰信号统计特性已知基础上的。对于一个实际系统，往往存在着模型不确定性或干扰信号统计特性不完全已知。这些不确定因素使得传统的 Kalman 滤波算法失去最优性，估计精度大大降低，严重时会引起滤波发散。近些年，人们将鲁棒控制的思想引入到滤波中来，形成了鲁棒滤波理论，比较有代表性的是 H_∞ 滤波。

信息融合和神经网络也有其他的许多优点，它们和 Kalman 滤波的结合在控制和估计领域内也同样是一个重要的发展方向。

以上介绍了 Kalman 滤波的发展过程，相信随着科技的不断发展进步，其理论将不断完善，应用领域将更加广泛。

1.4 Kalman 滤波的应用领域

一般地，只要跟时间序列和高斯白噪声有关或者能建立类似的模型的系统，

都可以利用 Kalman 滤波来处理噪声问题，都可以用其来预测、滤波。Kalman 滤波主要应用领域有以下几个方面。

（1）导航制导、目标定位和跟踪领域。

（2）通信与信号处理、数字图像处理、语音信号处理。

（3）天气预报、地震预报。

（4）地质勘探、矿物开采。

（5）故障诊断、检测。

（6）证券股票市场预测。

第 2 章　MATLAB 仿真基础

MATLAB 是美国 MathWorks 公司出品的商业数学软件，用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言和交互式环境，主要包括 MATLAB 和 Simulink 两大部分。经过多年的发展和多个版本的升级，如今 MATLAB 的功能已经非常强大了，它是当今最流行的计算机仿真软件之一。

有一定编程基础的读者可以跳过本章的学习，直接学习第 3 章。

2.1　MATLAB 简介

2.1.1　MATLAB 发展历史

MATLAB 的产生是与数学计算紧密联系在一起的。在 1980 年，美国新墨西哥州大学计算机系主任 Cleve Moler 在给学生讲授线性代数课程时，发现学生在高级语言编程上花费很多时间，于是着手编写供学生使用的 Fortran 子程序库接口程序，取名为 MATLAB（MATrix LABoratory 的前三个字母的组合，意思为“矩阵实验室”）。这个程序获得了很大的成功，受到学生的广泛欢迎。

20 世纪 80 年代初，Moler 等一批数学家与软件专家组建了 MathWorks 软件开发公司，继续从事 MATLAB 的研究和开发。1984 年推出第一个 MATLAB 商业版本，其核心是用 C 语言编写的。然后，MATLAB 又增加了丰富多彩的图形图像处理、多媒体、符号运算，以及其他流行软件的接口功能，至此 MATLAB 的功能逐渐强大。

具有划时代意义的是在 1992 年，MathWorks 公式正式推出 MATLAB 1.0 版本，到了 1999 年 MATLAB 5.3 版本进一步改进了原有功能，同时 Simulink 3.0 版本也达到较高水准。在 2000 年 10 月，MATLAB 6.0 版本推出，其无论是在操作界面，还是在程序发布窗口、历史信息窗口和变量管理窗口上，在操作和使用上都给用户提供了极大的方便。2001 年，MathWorks 公式又推出了 MATLAB 6.1 版/Simulink 4.1 版，其虚拟现实工具箱给仿真结果在三维视景下显示带来了新的解决方案。2003

年 6 月推出了 MATLAB Release 13，即 MATLAB 6.5/Simulink 5.0，在核心数值算法、界面设计、外部接口和应用等诸多方面有极大改进。2004 年正式推出 MATLAB Release 14，即 MATLAB 7.0/Simulink 6.0，这是一个具有里程碑意义的版本。此后，几乎每年的 3 月和 9 月 MathWorks 公司都会推出当年的 a 版和 b 版。目前的最新版本是 MATLAB 2014a。

MATLAB 是目前国际上最流行的科学计算与工程仿真软件工具之一，现在的 MATLAB 已经不仅仅是过去的“矩阵实验室”了，它已经成为具有广泛应用前景的、全新的计算机高级语言，可以说它是“第四代”计算机语言。自 20 世纪 90 年代以来，美国和欧洲各国家已将 MATLAB 正式列入研究生和本科生的教学计划，MATLAB 软件已经成为应用代数、自动控制理论、数理统计、数字信号处理、时间序列分析和动态系统仿真等课程的基本教学工具，成为学生所必须掌握的基本软件之一。在研究所和工业界，MATLAB 也成为工程师们必须掌握的一种工具，被认为是进行高效研究与开发的首选软件工具。

2.1.2　MATLAB 7.1 的系统简介

自 MATLAB 6.5 版本以后，各版本之间的界面风格都很相似，操作原理也是大同小异的。无论你采用哪一本版本，使用及操作都不会有很大的变化。本书采用的是 2005 年 9 月发布的 MATLAB 7.1 版本。在安装过程中，如果操作系统是 Windows XP，那会很顺利；如果用的是 Windows Vista 及更高版本的操作系统，那么在安装好并运行 MATLAB 软件时，会出现如下启动错误，如图 2.1 所示。

图 2.1　运行 MATLAB 软件时的系统启动错误

这时候请右击桌面的快捷图标，或者单击“开始”→“所有程序”中找到

MATLAB 后单击“属性”，弹出如图 2.2 所示对话框，勾选“以兼容模式运行这个程序”，在下拉列表框中选择 Windows Vista 即可，最后单击“确定”按钮。重新运行后能正常启动和使用。

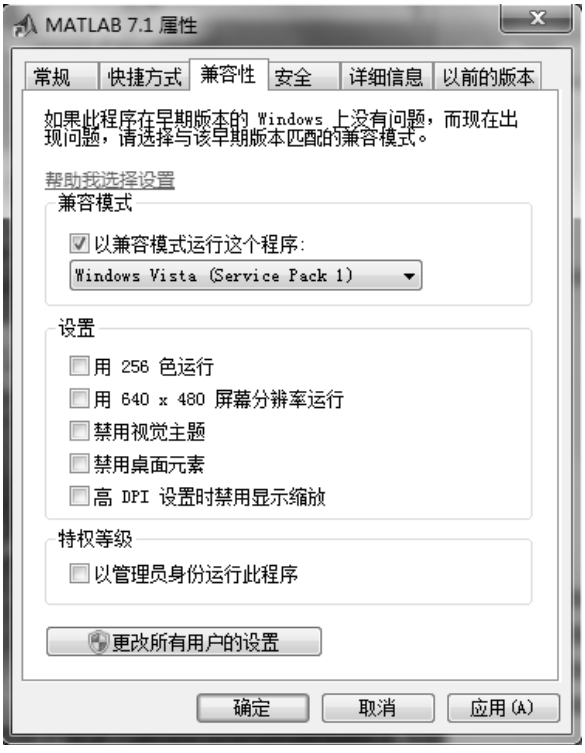


图 2.2　兼容模式设置

MATLAB 7.1 的系统界面如图 2.3 所示，它与之前的 7.0 版、6.5 版系统界面相差无几。系统界面主窗口中包括主菜单、工具栏、当前目录（Current Directory）窗口、工作空间（Workspace）窗口、命令历史（Command History）窗口等。最核心的则是命令窗口（Command Window），它在界面的右下侧。经常地，在移动各窗口时候会发生窗口布局紊乱，如果想恢复系统默认布局，可以单击主菜单 Desktop→Desktop Layer→Default 实现恢复默认窗口布局。

在命令窗口中，可以执行 MATLAB 的语句指令，例如想得到 $\sin(\pi/4)$ 的值，可以在命令窗口中输入 sin(pi/4)，可以得到结果如下。

```
>> sin(pi/4)
ans = 0.7071
```

图 2.3　MATLAB 7.1 的系统界面

另外，如果遇到任何不懂的函数，可以直接在命令窗口输入 help 查看该函数的功能和实例。通过 help 查阅 MATLAB 的文档说明，是每一个初学者快速掌握 MATLAB 有效方法。例如，在做目标跟踪时候，噪声符合伽马分布，那么如何用这个伽马分布的函数呢？可以在命令窗口中输入 help gamrnd，得到图 2.4 所示帮助。

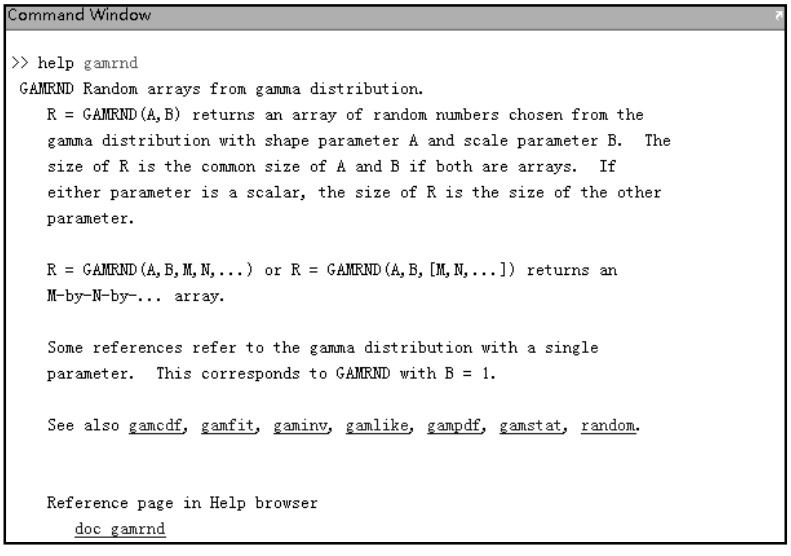


图 2.4　help 帮助系统

如果想再进一步查看它的使用实例，可以单击 doc gamrnd，则有关于该函数的使用例子，如图 2.5 所示。

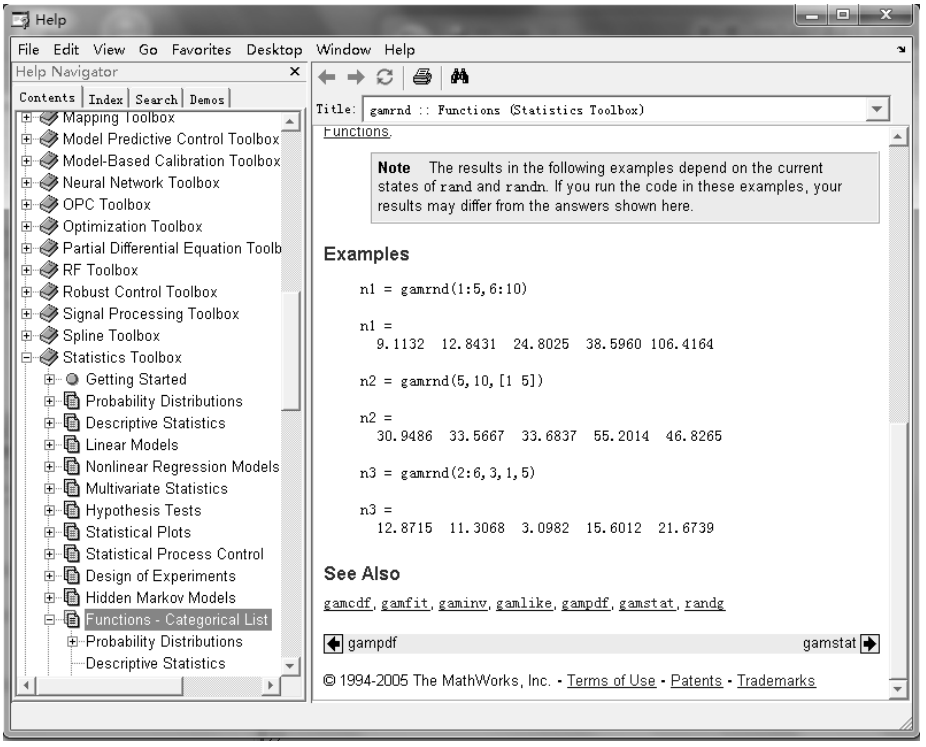


图 2.5　help 文档说明

2.1.3　M 文件编辑器的使用

当用户要运行的指令较多的时候，总不能一直在命令窗口调试，那样无法修改，且太费时间了。而 MATLAB 的 M 文件编辑器能解决这一问题，用户可以将一组相关的指令编辑在同一个 ASCII 码命令文件中，即所谓的编程。在主菜单下面，第一个快捷工具 即为新建一个 M 文件，单击它可以建立一个未命名的 M 文件，另外也可以通过单击主菜单 File→New→M-File 方式建立一个新的文档，如图 2.6 所示。M 文件编辑器包含主菜单、工具栏、代码编辑窗口等。

现在，开始写第一个 MATLAB 程序。按照上述方法，创建一个 M 文件，在代码编辑窗口中输入以下程序。

```
%%%
```

```
% 数值计算，语句结束不加分号，可以在命令窗口查看结果
value=sin(pi/4)+cos(1)+log(2)
% 在命令窗口输出文本
disp('Hello World');
% 画一个正弦函数
t=0:0.5:2*pi;
y=sin(t);
plot(t,y,'-ko','MarkerFace','g')
```

图 2.6　M 文件编辑器

代码输入完毕，按 Ctrl+S 键或者单击 M 编辑器的“保存”按钮，把 M 文件名写为 example1_1.m，然后单击工具栏中的运行程序（Run）按钮，运行结果如图 2.7 所示。

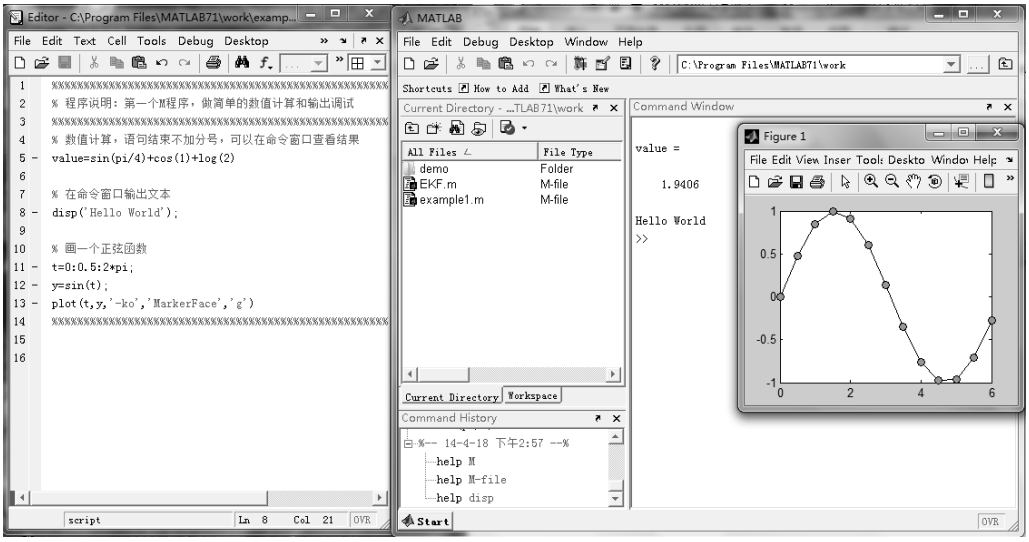


图 2.7　程序运行结果

2.2 数据类型和数组

本节重点介绍 MATLAB 内置的数据操作方法，首先介绍数据的类型，接着重点介绍数值类型，然后介绍 MATLAB 最重要的数组。数组的概念和操作是本节的重点，希望读者熟练掌握。

2.2.1 数据类型概述

MATLAB 中有 15 种基本的数据类型，分别是 8 种常规数据类型（int8、uint8、int16、uint16、int32、uint32、int64、uint64）、单精度浮点数、双精度浮点数、逻辑数据类型、字符串类型、元胞数组、结构体、函数句柄，见表 2.1。另外，为了和高级语言交叉编译，MATLAB 有用户自定义的面向对象的用户类类型和 Java 类类型。

表 2.1 MATLAB 中的数据类型

数 据 类 型	示 例	说 明
int8、uint8 int16、uint 16 int32、uint 32 int64、uint 64	a=uint16(8000) b=int8(123.5)	（1）有符号和无符号的整数类型。 （2）大部分整数类型占用比浮点类型更少的内存空间。 （3）除了 int64 和 uint64 类型外的所有整数类型，都可以用在数学计算中
single	single(383.21)	（1）单精度浮点类型。 （2）和双精度类型相比，占用内存空间少
double	123.45 4+1.234i	（1）双精度浮点类型。 （2）它比单精度浮点类型表示范围广，是 MATLAB 默认的数值类型
logical	a=randn>0.5	（1）逻辑数据类型。 （2）如果 randn 的值大于 0.5，则 a 得到的是逻辑值 1，否则为 0
char	‘Hello World !’	字符串类型
cell array	A{1,1}=’Jack’ A{1,2}=25 A{1,3}=[1,2;3,4]	（1）元胞数组类型。 （2）数组元素可以是不同的数据类型
structure	station.id=123 station.x=100 station.y=120	（1）结构体类型。 （2）有 C 语言基础的读者不难理解，它与 C 语言中的结构体类似，该结构体成员可以存储多种类型的数据
函数句柄	@sin	函数句柄，相当于一个指针

这 15 种基本的数据类型都是按照数组的形式在内存中存储和操作的。8 种常规数据类型，可以简单概括为“整数”，单精度和双精度浮点类型数据可以简单概括为“小数”。在这两种数之间有时会出现互相转换，下面通过几个例子说明它们

之间的转换。

【例 2.1】 将浮点数 128.4 转换为整数。

方法 1：a=int8(128.4)，结果为 a=127，高位溢出。因为 128.4 超出了 int8 的表示范围（$-2^7 \sim 2^7$），这时候请用 a=int16(128.4)，则 a=128。

方法 2：利用向最接近的整数靠近取整 round()函数，即 a=round(128.4)，得到 a=128。因为小数部分是 0.4<0.5，用该函数时小数部分大于等于 0.5 时则舍弃小数部分，整数部分加 1，如 round(128.5)=129。

方法 3：利用向 0 取整函数 fix(x)，即 a=fix(128.4)，结果为 a=128。fix(−128.4)=−128。

方法 4：利用向不大于 x 的最接近整数取整 floor(x)。floor(128.4)=128，floor(−128.4)=−129，请注意其与 fix(x)函数的区别。

方法 5：利用向不小于 x 的最接近整数取整 ceil(x)。ceil(128.4)=129，ceil(−128.4)=−128。请注意其与 floor(x)的对比。

2.2.2　数组的创建

按照数组元素的个数和排列方式，MATLAB 中的数组可以分为以下几种。

（1）没有元素的空数组（empty array）。

（2）只有一个元素的标量（scalar），它实际上是一行一列的数组。

（3）只有一行或者一列元素的向量（vector），分别叫做行向量和列向量，也统称为一维数组。

（4）普通的具有多行多列的二维数组。

（5）超过二维的多维数组。

【例 2.2】 创建一个空数组 A。

```
>> A=[]   % 矩阵右边是一个中括号包围的空矩阵
```

【例 2.3】 创建一个一维的数组 A。

方法 1：A=[1,2,3,4,5]

方法 2：A=ones(1,5)，结果为

```
A =
     1     1     1     1     1
```

方法 3：通过冒号来创建。

```
>> A = 1:5
A =
     1     2     3     4     5
```

【例 2.4】 创建一个二维数组 A。

方法 1：A = [1 2 3;4,5,6;7 8 9]。注意元素之间可以用空格隔开也可以用逗号隔开，同时分行符号必须用分号。

方法 2：A=zeros(3,3)、A=ones(3,3)等。创建一个 3 行 3 列的，每个元素为 0 或者 1 的元素。

方法 3：A=[1:3;linspace(4,6,3);7 8 9]，在命令窗口的运行结果如下。

```
A =
     1     2     3
     4     5     6
     7     8     9
```

冒号的作用，如 1:N 它表示的意思是从 1 开始，每次增加 1 直到 N。若指定增长步长，如 1:2:N，则表示从 1 开始每次增加 2，一直到小于等于 N 的最大整数。例如，在命令输入 1:2:10 得到以下结果。

```
>> 1:2:10
ans =   1     3     5     7     9
```

linspace(start,stop,n)函数的作用是产生一个从 start 开始的，最后一个是 stop 的等差数列，公差为(stop- start)/(n-1)。

【例 2.5】 创建一个三维数组。

创建三维或者更高维的数组，无法直接赋值得到，需要通过指定索引把二维数组扩展成多维的，或者通过 MATLAB 内部的函数，如 ones()、zeros()、cat()等。

方法 1：通过索引将二维数组扩展成多维数组。

```
A=zeros(2,2,3)          % 定义一个三维数组 A 并将其所有元素初始化为 0
A(:,:,1)=[1,2;3,4]      % 将阵列 1 赋值一个 2 行 2 列的矩阵
A(:,:,2)=[5,6;7,8]      % 将阵列 2 赋值一个 2 行 2 列的矩阵
A(:,:,3)=[9,10;11,12]   % 将阵列 3 赋值一个 2 行 2 列的矩阵
```

在这里，冒号所在的位置表示该维的全部索引。例如，A(:,:,1)表示阵列 1 的所有行（2 行）和所有列（2 列）。

方法 2：用 cat()函数创建。

```
A=[1,2;3,4];
B=[5,6;7,8];
C=cat(3,A,B) %按照第 3 维将 A 和 B 连接起来
```

最后，读者请注意，在创建任何一个数组时，建议先用 zeros()函数初始化，然后对各元素值赋值。

2.2.3　数组的属性

MATLAB 中提供了大量的函数，用于返回数组的各种属性，包括数组的排列结构、数组的尺寸大小、维度、数组数据类型，以及数组在内存中占用的空间情况等等。本节通过例子重点介绍数组的尺寸大小和维度的获取方法。

【例 2.6】 获取任意一个数组的大小 size()函数。

```
>> A=[]; %  空数组
>> size(A)
ans =
     0     0              %  表示 0 行 0 列
>> B=[1,2,3;4,5,6];       %  2 行 3 列的数组
>> size(B)
ans =
     2     3              %  size()的返回值，表示 2 行 3 列
```

可见 size(A)函数得到是数组的行和列。另外也可以用 length(A)函数，当 A 是一维数组时，length(A)返回的是 A 数组的元素个数；当 A 是二维数组时候，length(A)返回 size(A)得到的行和列中较大的那个值。

【例 2.7】 用 length 获得矩阵 A 的每一维度的元素个数。

```
>> A=randn(3,4,5,2);          %  用随机函数产生了一个 4 维数组
>> n1=length(A(:,4,5,2))      %  返回第 1 维的长度
n1 = 3
>> n2=length(A(3,:,5,2))      %  返回第 2 维的长度
n2 = 4
>> n3=length(A(3,4,:,2))      %  返回第 3 维的长度
n3 = 5
>> n4=length(A(3,4,5,:))      %  返回第 4 维的长度
n4 = 2
```

【例 2.8】 获得数组的维度。

空数组、单个元素、一维数组，在 MATLAB 里都将其视为二维数组，因为它们都至少具有两个维度（至少具有行和列两个方向）。获取数组的维度用 ndims() 函数如下。

```
>> A=[];
>> ndims(A)              % A 为空数组，返回为 2，可见至少是二维的
ans =2
>> A=zeros(2,3);         % A 为二维数组，返回值为 2
>> ndims(A)
ans =2
>> A=randn(2,2,3);       % A 为三维数组，返回值为 3
>> ndims(A)
ans =3
```

2.2.4 数组的操作

数组的操作主要有对数组的索引和寻址，数组的裁剪和元素删除，数组形状的改变，数组的运算，以及数组元素的排序等。下面通过例子来讲解数组的主要操作方法。

【例 2.9】 对数组元素的索引与寻址。

```
>> A=rand(3,4)     % 用随机分布函数产生一个 3 行 4 列的数组
A =
    0.9501    0.4860    0.4565    0.4447
    0.2311    0.8913    0.0185    0.6154
    0.6068    0.7621    0.8214    0.7919
>> A(2,3)           % 双下标索引访问的数组的第 2 行第 3 列的元素
ans =0.0185
>> A(3)             % 单下标索引第 3 个元素（即第 3 行第 1 列元素），默认为列方向开始
ans =0.6068
>> A(4)             % 单下标索引第 4 个元素（即第 1 行第 2 列元素），默认为列方向
ans =0.4860
>> A(1:2)           % 单下标索引第 1 到第 2 个元素，默认为列方向
ans =0.9501    0.2311
>> A(2,1:3)         % 双下标索引第 2 行，第 1 到 3 列元素
ans =0.2311    0.8913    0.0185
>> A(:,[1,3,4])     % 双下标索引，所有行的第 1、3、4 列
```

```
ans =
    0.9501    0.4565    0.4447
    0.2311    0.0185    0.6154
    0.6068    0.8214    0.7919
```

【例 2.10】 对数组元素的裁剪和删除。

```
>> A=magic(6)   % 产生 6*6 的魔方数组
A =
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
>> B=A(1:2,1:2:5)   % 提取数组 A 第 1 到 2 行的 1、3、5 列赋给 B
B =
    35     6    19
     3     7    23
>> C=A(1,[2,4,6])   % 将数组 A 的第一行中第 2、4、6 列元素赋给 C
C =
     1    26    24
>> D=A(3:8)    % 将数组 A 的第 3 到 8 元素赋给 D
D =
    31     8    30     4     1    32
>> A([2,4,5,6],:)=[]    % 将 A 数组的 2、4、5 、6 行删除
A =
    35     1     6    26    19    24
    31     9     2    22    27    20
>> B=[1 2;3 4];
A=[A B]          % 数组的扩展，将 B 数组添加到 A 数组的后面
A =
    35     1     6    26    19    24     1     2
    31     9     2    22    27    20     3     4
```

【例 2.11】 数组的转置。

MATLAB 中进行数组转置最简单的是通过转置操作符（'）。

```
>> A=[1 2;3 4]
>> A'
```

```
ans =
     1     3
     2     4
```

【例 2.12】 数组的加减乘除运算。

```
>> A=[1 2 3;4 5 6;7 8 9]
>> B=diag([3 2 1])
>> A-B   % 数组的减法运算
ans =
    -2     2     3
     4     3     6
     7     8     8
>> A+B    % 数组的加法运算
ans =
     4     2     3
     4     7     6
     7     8    10
>> A*B   % 数组的乘法运算
ans =
     3     4     3
    12    10     6
    21    16     9
>> A^3   % 数组的乘方运算
ans =
   468   576   684
  1062  1305  1548
  1656  2034  2412
>> A/B        % 数组的除法运算
ans =
    0.3333    1.0000    3.0000
    1.3333    2.5000    6.0000
    2.3333    4.0000    9.0000
>> A*inv(B)   % 与 A/B 运算是等价的
ans =
    0.3333    1.0000    3.0000
    1.3333    2.5000    6.0000
    2.3333    4.0000    9.0000
```

【例 2.13】 数组的排序。

```
>> A=rand(1,5)
A =
    0.4447    0.6154    0.7919    0.9218    0.7382
>> sort(A)
ans =
    0.4447    0.6154    0.7382    0.7919    0.9218
```

默认情况下，sort()函数对数组是按照升序排列。读者可以利用 help sort 查看 sort()函数的其他排序方式的使用方法。

数组是 MATLAB 中各种变量存储和运算的通用数据结构。希望读者重点掌握，这对今后的编程非常有帮助。

2.2.5 结构体和元胞数组

本节介绍 MATLAB 中两种复杂的数据类型：结构体（Structure）和元胞数组（Cell Array）。这两种类型类似数组，都可以存储不同类型的数据，在程序中应用广泛。本节主要介绍这两种数据类型的创建、内部数据的索引寻址以及与其相关的操作函数。

1．结构体

结构体的创建有两种方法：直接采用赋值语句给结构体的字段赋值；通过结构体创建函数来创建。

【例 2.14】 通过对字段赋值创建结构体。

```
station.name='s1';
station.x=100;
station.y=120;
```

通过“结构体名称.字段名称”的形式对结构体创建和赋值。例 2.14 中创建了一个基站（station）结构体，并将名称（name）字段赋值为's1'，将基站的坐标（x,y）设为（100,120）。同理，可以创建结构体数组，如下。

```
station(1).name='s1'，station(1).x=100，station(1).y=120;
station(2).name='s2'，station(2).x=101，station(2).y=121;
station(3).name='s3'，station(3).x=102，station(3).y=123;
```

这里采用对结构体数组分别赋值法，创建一个含有 3 个元素的结构体数组，每个结构体对象都有名称和坐标属性。如果要获取它们的数值，如要得到其中一个结构体的 x 坐标，可以将其直接赋给某个变量，如下。

```
xx= station(1).x
```

【例 2.15】 通过 struct 创建结构体。

```
>> StationGroup=struct('name',{'s1','s2','s3'},'x',{100,101,102},
'y',{120,121,122})
```

struct(字段名称,字段值,字段名称,字段值……)，通过该方法创建了一个结构体数组。通过下标索引的方式访问其中一个成员，例如：

```
>> StationGroup(1)
ans =
      name: 's1'
         x: 100
         y: 120
```

【例 2.16】 结构体的嵌套。

```
station.position.x=10;
station.position.y=11;
station.id.newid.n=3;
```

可见结构体可以有多个字段，每个字段也可以继续成为结构体，这就是结构体的嵌套。

2．元胞数组

创建元胞数组可以通过直接赋值法和 cell 函数法。在元胞数组中，经常要用到花括号{}。它有两种使用方法。

（1）花括号用在下标索引上，则出现在赋值语句等号左侧，那么右侧只写索引表示的位置上元胞内的数据，例如：

```
>> A{1,1}=randn(2)     % 直接赋值法创建元胞数组
A =
      [2x2 double]
>> A{1,2}=randn(3)    % 直接赋值法创建元胞数组
```

```
A =
    [2x2 double]    [3x3 double]
>> B=cell(2,2)      % cell 函数法创建元胞数组，相当于创建 2×2 个数据块
B =
     []     []
     []     []
>> B{1,1}=randn(2)  % 索引元胞数组，并对其重新赋值（第 1 数据块）
B =
    [2x2 double]     []
                []     []
```

（2）元胞数组左边是小括号，那么在赋值时等号右边必须用花括号，如果赋值的元素是数组，需要用中括号，例如：

```
>> A(1,1)={1}    % 注意等号右边为花括号，单个元素可以不用中括号
A =
    [1]
>> A(1,2)={[1 2]}  % 含有多个元素，需要用中括号，表示为一个数组
A =
    [1]    [1x2 double]
>> value=A{1,2}(1,1)     % 索引元胞数组 A 的元素，与数组相似
value =1
```

2.3　程序设计

MATLAB 是一种高效的编程语言，和其他高级语言一样，MATLAB 也提供了循环语句、条件转移语句等一些常规的控制语句，而且与 C 语言的控制语句很相似。

与程序流程控制有关的 MATLAB 关键字有 if、else、end、switch、case、otherwise、for、while、continue、break 等。能熟练掌握这些关键字，对于 MATLAB 编程至关重要。

在 MATLAB 中，注释为“%”，用在任意一条语句后，例如：

```
C=3;  % 这是注释，如果将分号去掉，C 的值会显示在命令窗口
```

2.3.1　条件语句

条件语句主要有 if、switch 语句。if 语句基本形式是 if-else-end，if 语句中可

以嵌套多个 elseif 语句，常用的 if 语句的格式如下。

```
if 条件表达式 1
    分支语句 1
elseif 条件表达式 2   （elseif 可选）
    分支语句 2
else
    分支语句（默认）
end
```

当条件表达式 1 为真时，执行分支语句 1（否则查看是否满足条件表达式 2，如果该表达式为真，则执行分支语句 2），如果条件表达式为假，则执行默认的分支语句，最后结束。下面举例说明其具体的使用。

【例 2.17】 计算 $f(x)=\begin{cases}1 & x<-\pi \\ 3x & x>\pi \\ \sin x & -\pi \leqslant x \leqslant \pi\end{cases}$

```
x=10,fx=0;    % x 可以设置为任意值
if x<(-1)*pi
    fx=1
elseif x>pi
    fx=3*x
else
    fx=sin(x)
end
```

switch 与 case 配合使用。值得注意的是，在 MATLAB 中的 switch 表达式可以是字符串，其语句的格式如下。

```
switch 表达式（标量或字符串）
    case 值 1
        语句 1
        ……
    case 值 n
        语句 n
    otherwise
        默认语句
end
```

【例 2.18】 输入 2014 年的某个月份，输出该月份的天数。

```
n=4   % 输入 4 月份
switch(n)
    case 2
        result=28
    case 4
        result=30
    case 6
        result=30
    case 11
        result=30
    otherwise
        result=31
end
```

switch 语句中各分支结束无需用 break 关键词。这点与 C 语言不同，请读者注意。

2.3.2　循环语句

MATLAB 中的循环语句包括 for 循环和 while 循环两种类型。for 循环的基本格式如下。

```
for 循环变量=起始值:步长:终止值
    循环体
end
```

步长默认值为 1，步长可以在正实数或负实数范围内任意指定。对于正数，循环变量的值大于终止值时，循环结束；对于负数，循环变量的值小于终止值时，循环结束。

【例 2.19】 计算 $\text{sum}=1+2+3+\cdots N$，$N$=10。

```
sum=0
for n=1:10
    sum=sum+n
end
```

while 循环的格式如下。

```
while 表达式
    循环体
end
```

其执行方式为，如果表达式为真，则执行循环体的内容，执行后再判断表达式是否为真，若为假则跳出循环体，向下继续执行，否则跳出结束。

【例 2.20】 计算 $\text{sum}=1+2+3+\cdots N$，当 sum>100 时停止。

```
sum=0;n=0; %初始化
while sum<=100
    n=n+1
    sum=sum+n
end
```

程序在 n=14 时结束，这时 sum=105。

在循环中，常常会用到 continue 和 break 语句。continue 语句表示当次循环不再继续向下执行，而是直接对循环变量进行递增，进入下一次循环。而 break 语句用于退出循环。

【例 2.21】 从 100 个随机的整数（大小范围是 0～50）中挑出大于 25 的数，并对它们求和，当和大于 150 时可以停止，并打印出挑出的整数。

```
%用 randint 函数产生 1 行 100 列，大小在 0～50 之间的随机整数
A=randint(1,100,[0 50]);      % 调用 randint 函数产生 100 个随机的整数，并赋给 A
sum=0;
B=[];                         % 用于存放大于 25 的数的数组
for i=1:100
    if A(i)<=25
        continue;             % 小于 25 的数，继续下一轮
    else
        sum=sum+A(i);         % 对于大于 25 的数求和
        B=[B A(i)];           % 对于大于 25 的数插在 B 数组的后面，保存
    end
    if sum>150
        break;                % 如果和大于 150，则终止循环
    end
end
% 以下语句不加分号，可以在命令窗口看运行结果
sum
B
```

运行结果如下。

```
sum =    161
B =    48      30      45      38
```

2.3.3　函数

和其他高级语言一样，MATLAB 中的函数接收输入参数（也可无输入参数），返回输出参数（也可无返回值），定义函数的关键字是 function。定义函数的格式如下。

```
function   [ 输出参数 1,输出参数 2……] =
函数名(输入参数 1,输入参数 2,……)
```

建议在书写函数时，函数名与 M 文件名保持一致。例如，我们书写一个主函数，调用 M 文件编辑器，写入如图 2.8 所示代码，保存时，会默认 M 文件名为 main。子函数可以与主函数写在同一个 M 文件中，也可以单独保存为 M 文件。但是在运行时，一定要将它们放在同一工作目录下。

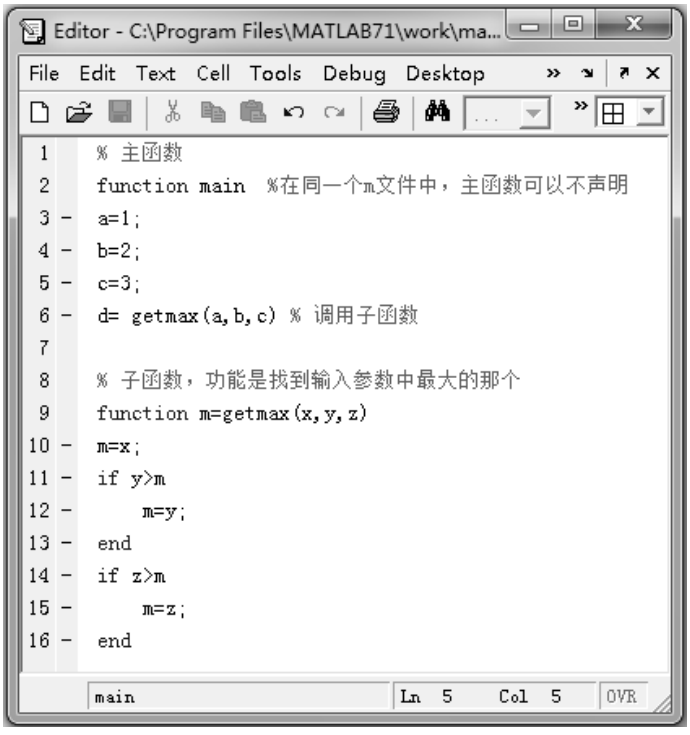


图 2.8　主函数和子函数

MATLAB 中有一种函数叫匿名函数，它通常是一行代码能写完的简单函数。与 M 文件一样，匿名函数可以接收多个参数，创建匿名函数的格式如下。

```
fhandle=@ (参数列表) 表达式
```

符号@是 MATLAB 中创建函数句柄的操作符，表示创建由输入参数列表和表达式确定的函数句柄，并把这个函数句柄返回给变量 fhandle，这样就可以通过 fhandle 来调用定义好的这个函数了。例如：

```
>> myfun=@(x,y)(x+y^2)
myfun =
    @(x,y)(x+y^2)
>> myfun(1,2)
ans = 5
```

函数句柄实际上提供了一种间接调用函数的方法。MATLAB 提供的各种 M 文件函数和内部函数，都可以创建函数句柄，通过函数句柄对这些函数实现间接调用。创建函数句柄的一般语法格式如下。

```
fhandle=@function_filename
```

其中，function_filename 是函数对应的 M 文件的名称或者 MATLAB 内部函数的名称，@是句柄创建操作符，fhandle 是保存函数句柄的变量。例如，fhandle=@sin 就创建了 MATLAB 内部函数 sin 的句柄，并保存在 fhandle 变量中，以后就可以通过 fhandle(x)来实现 sin(x)的功能。读者也可以编写自己的函数，如图 2.9 所示，自

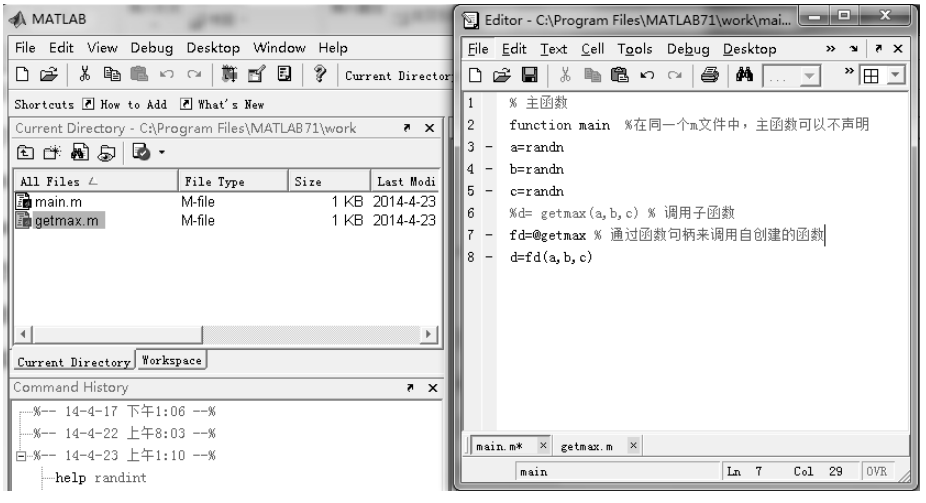


图 2.9　函数句柄的创建和调用

定义的子函数为 getmax(x,y,z)（函数体与图 2.8 实现内容一致，只是将它单独保存在 M 文件中），通过 fd=@getmax 实现句柄创建，并用 fd(x,y,z)实现调用。

2.3.4　画图

MATLAB 中有各种画图函数可供读者调用，例如 plot、plot3、bar、line 等。下面以 plot 函数为例，介绍如何利用其绘制各种不同的图形。plot 函数的语法格式如下。

```
plot(X1,Y1,LineSpec,……)
```

可以通过字符串 LineSpec 指定曲线的线型、颜色以及数据点的标记类型。这在突出显示原始数据点和个性化区分多组数据的时候是十分有用的。

例如，“-.or”就表示采用点画线，数据点用圆圈标记，颜色是红色。MATLAB 默认是用颜色区分多组曲线的，但在只能黑白打印或者显示的情况下，个性化设置曲线的线型就成为唯一的区分方法了。

表 2.2 列出了 MATLAB 可供选择的曲线的线型、颜色和数据点标记类型，这对于其他 MATLAB 画图函数都是通用的。

表 2.2　LineSpec 可选字符列表

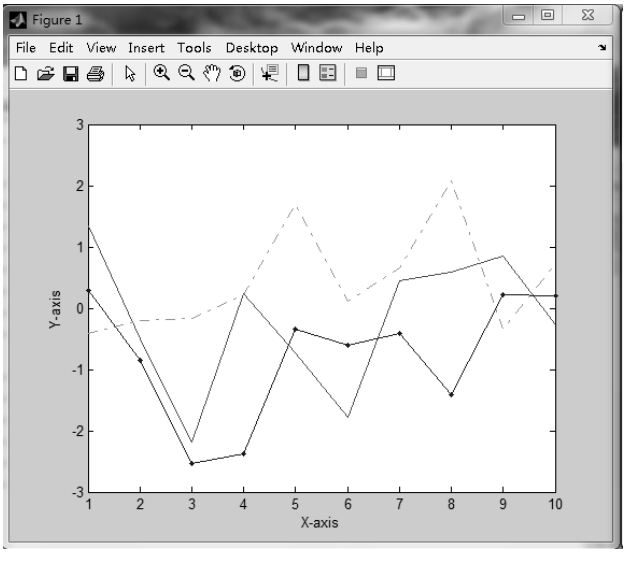
线型		颜色		数据点标记类型	
标识符	意义	标识符	意义	标识符	意义
-	实线	r	红色	+	加号
-.	点画线	g	绿色	o	圆圈
--	虚线	b	蓝色	*	星号
:	点线	c	蓝绿色	.	点
		m	洋红色	x	交叉符号
		y	黄色	s（或者 squre）	方形
		k	黑色	d（或者 diamond）	菱形
		w	白色	^	向上的三角形
				v	向下的三角形
				>	向右的三角形
				<	向左的三角形
				p（或者 pentagram）	五边形
				h（或者 hexagram）	六边形

【例 2.22】 通过用随机函数 randn 产生 3 组随机数，每组 10 个，将数据用 plot

画出，并设置不同的线性和颜色。

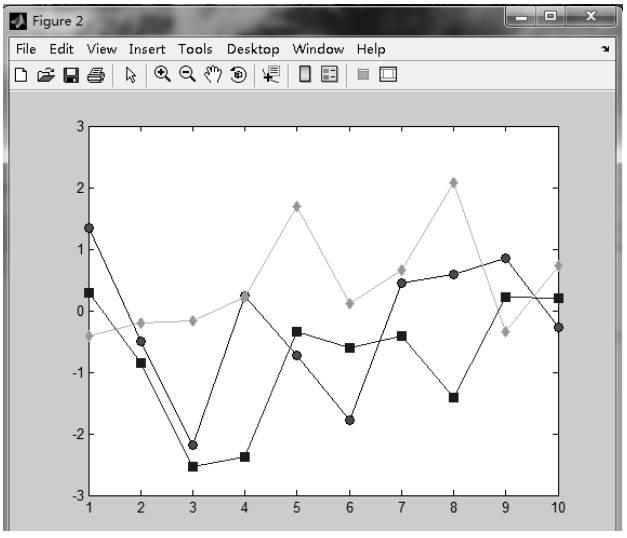
```
function main   % 主函数
A1=randn(1,10);
A2=randn(1,10);
A3=randn(1,10);
% 画图 1
figure
box on
hold on; %在同一个 figure 中多次调用 plot，需要 hold
plot(A1,'-r') %红色的实线
plot(A2,'-.g') %绿色的点画线
plot(A3,'-b.') %蓝色的实线，数据点为黑实点
xlabel('X-axis')
ylabel('Y-axis')
% 画图 2
figure
box on
hold on; %在同一个 figure 中多次调用 plot，需要 hold
plot(A1,'-ko','MarkerFaceColor','r') %黑色实线，红色圆圈数据点
plot(A2,'-cd','MarkerFaceColor','g') %蓝绿色实线，绿色菱形数据点
plot(A3,'-bs','MarkerFaceColor','b') %蓝色实线，蓝色方形数据点
```

程序运行结果如图 2.10 所示。



（a）

图 2.10　设置不同颜色和线型的画图结果



（b）

图 2.10　设置不同颜色和线型的画图结果（续）

读者可以尝试不同的组合，来画出各种精美的图形效果。同时读者还可以在 plot 绘图的同时设置曲线的线宽、标记点的大小、标记点内的填充颜色等。这些都是通过 plot(…,'PropertyName', PropertyValue,…)这样的语法格式来实现的，请参照表 2.3 的参数说明。

表 2.3　绘图命令中可选 PropertyName

PropertyName	意　义	选　项
LineWidth	线宽	数值，如 0.5、1、2.5 等
MarkerEdgeColor	标记点边框线条颜色	颜色字符，如 r、g、b
MarkerFaceColor	标记点内部填充颜色	颜色字符，如 r、g、b
MarkerSize	标记点大小	数值，如 0.5、1、2.5 等

2.4　小结

本章从介绍 MATLAB 发展历史入手，简单介绍了 MATLAB 7.1 版本的系统。2.2 节介绍了 MATLAB 的数据类型，读者要重点掌握数组的概念和操作方法，在编程用应该能熟练使用数组操作的各种小技巧。2.3 节介绍了程序设计方法，主要有分支和循环的程序流程控制语句，接着介绍了函数的使用方法，最后是可视化绘图。MATLAB 强大的数据可视化功能，能为程序仿真提供各种美观的输出。

本章是 MATLAB 在 Kalman 滤波仿真应用中的基础，希望初学者认真掌握。同时希望读者能参阅其他介绍 MATLAB 编程的书籍，毕竟本书的重点不是介绍如何使用 MATLAB 编程，希望读者理解。

第 3 章　线性 Kalman 滤波

在许多工程实践中，往往不能直接得到所需要的状态变量的真实值。例如，雷达在探测空中目标的时候，根据反射波等信息能算出目标的距离，但是雷达探测过程中存在随机干扰的问题，导致在观测得到的信号中往往夹杂有随机噪声。我们要从夹杂有随机噪声的观测信号中分离出飞机或导弹的运动状态量，要准确地得到所需的状态变量是不可能的，只能根据观测信号来估计或预测这些状态变量。Kalman 滤波器就是这种能有效降低噪声影响的利器。在线性系统中，Kalman 滤波是最优滤波器。随着计算机技术的发展，Kalman 滤波的计算要求与复杂性已不再成为其应用的障碍，它越来越受到人们的青睐。目前 Kalman 滤波理论已经广泛应用在国防、军事、跟踪、制导等许多高科技领域。

3.1　Kalman 滤波原理

在几何上，Kalman 滤波器可以看作状态变量在由观测生成的线性空间上的射影。因此射影定理是 Kalman 滤波推导的基本工具。在介绍线性离散系统的 Kalman 滤波方程之前，先介绍射影定理。

3.1.1　射影定理

【定义 3.1】 由 $m\times1$ 维随机向量 $\boldsymbol{y}\in R^m$ 的线性函数估计 $n\times1$ 维随机变量 $\boldsymbol{x}\in R^n$，记估值为

$$\hat{\boldsymbol{x}}=\boldsymbol{b}+\boldsymbol{A}\boldsymbol{y},\boldsymbol{b}\in R^n,\boldsymbol{A}\in R^{n\times m} \tag{3.1}$$

若估值 $\hat{\boldsymbol{x}}$ 极小化性能指标为 J，即

$$J=\mathrm{E}[(\boldsymbol{x}-\hat{\boldsymbol{x}})^{\mathrm{T}}(\boldsymbol{x}-\hat{\boldsymbol{x}})]$$

则称 $\hat{\boldsymbol{x}}$ 为随机变量 $\boldsymbol{x}$ 的线性最小方差估计，其中 E 为均值号，T 为转置号。

由观测值 $\boldsymbol{y}$ 求随机变量 $\boldsymbol{x}$ 的线性最小方差估计的表达式为

$$\hat{\boldsymbol{x}}=\mathrm{E}\boldsymbol{x}+\mathrm{Cov}(\boldsymbol{x},\boldsymbol{y})\mathrm{Var}(\boldsymbol{y})^{-1}(\boldsymbol{y}-\mathrm{E}\boldsymbol{y}) \tag{3.2}$$

线性最小方差估计 $\hat{\boldsymbol{x}}$ 具有如下性质。

（1）无偏性，即 $\mathrm{E}\hat{\boldsymbol{x}} = \mathrm{E}\boldsymbol{x}$。

（2）正交性，即 $\mathrm{E}[(\boldsymbol{x}-\hat{\boldsymbol{x}})\boldsymbol{y}^{\mathrm{T}}] = \boldsymbol{0}$。

（3）$\boldsymbol{x}-\hat{\boldsymbol{x}}$ 与 $\boldsymbol{y}$ 是不相关的随机变量。

【定义 3.2】 称 $\boldsymbol{x}-\hat{\boldsymbol{x}}$ 与 $\boldsymbol{y}$ 不相关为 $\boldsymbol{x}-\hat{\boldsymbol{x}}$ 与 $\boldsymbol{y}$ 正交（垂直），记为 $\boldsymbol{x}-\hat{\boldsymbol{x}} \perp \boldsymbol{y}$，并称 $\hat{\boldsymbol{x}}$ 为 $\boldsymbol{x}$ 在 $\boldsymbol{y}$ 上的射影，记为 $\hat{\boldsymbol{x}} = \mathrm{proj}(\boldsymbol{x} \mid \boldsymbol{y})$。

射影的几何意义如图 3.1 所示。

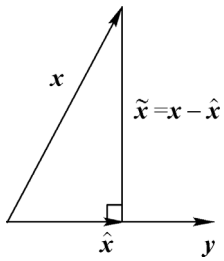


图 3.1　射影的几何意义

【定义 3.3】 基于随机变量 $\boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k) \in R^m$，对随机变量 $\boldsymbol{x} \in R^m$ 的线性最小方差估计 $\hat{\boldsymbol{x}}$ 定义为

$$\hat{\boldsymbol{x}} = \mathrm{proj}(\boldsymbol{x} \mid \boldsymbol{w}) \triangleq \mathrm{proj}(\boldsymbol{x} \mid \boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k)) \tag{3.3}$$

也称 $\hat{\boldsymbol{x}}$ 为 $\boldsymbol{x}$ 在线性流型 $L(w)$或 $L(\boldsymbol{y}(1), \cdots, \boldsymbol{y}(k))$ 上的射影。

【定义 3.4】 设 $\boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k) \in R^m$ 是存在二阶矩的随机序列，它的新息序列（新息过程）定义为

$$\boldsymbol{\varepsilon}(k) = \boldsymbol{y}(k) - \mathrm{proj}(\boldsymbol{y}(k) \mid \boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k-1))，\ k=1,2,\cdots \tag{3.4}$$

并定义 $\boldsymbol{y}(k)$ 的一步最优预报估值为

$$\hat{\boldsymbol{y}}(k \mid k-1) = \mathrm{proj}(\boldsymbol{y}(k) \mid \boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k-1)) \tag{3.5}$$

因而新息序列可定义为

$$\boldsymbol{\varepsilon}(k) = \boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k \mid k-1)，\ k=1,2,\cdots \tag{3.6}$$

式中，规定 $\hat{\boldsymbol{y}}(1 \mid 0) = \mathrm{E}\boldsymbol{y}(1)$，这保证了 $\mathrm{E}\boldsymbol{\varepsilon}(1) = \boldsymbol{0}$。新息 $\boldsymbol{\varepsilon}(k)$ 的几何意义如图 3.2 所示，可以看出 $\boldsymbol{\varepsilon}(k) \perp L(\boldsymbol{y}(1), \cdots, \boldsymbol{y}(k-1))$。

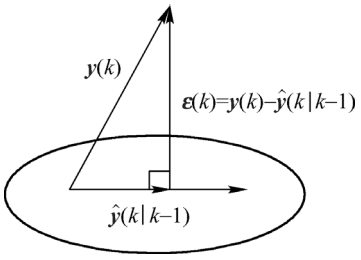


图 3.2　新息 $\boldsymbol{\varepsilon}(k)$ 的几何意义

【推论 3.1】 设随机变量 $\boldsymbol{x} \in R^n$，则有

$$\text{proj}(\boldsymbol{x} \mid \boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k)) = \text{proj}(\boldsymbol{x} \mid \boldsymbol{\varepsilon}(1), \boldsymbol{\varepsilon}(2), \cdots, \boldsymbol{\varepsilon}(k)) \tag{3.7}$$

由于新息序列的正交性，这一定理将大大简化射影的计算。

【定理 3.1】 （递推射影定理）设随机变量 $\boldsymbol{x} \in R^n$，随机序列 $\boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k) \in R^m$，且它们存在二阶矩，则有递推射影公式

$$\begin{aligned}\text{proj}(\boldsymbol{x} \mid \boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k)) = \text{proj}(\boldsymbol{x} \mid \boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k-1)) + \\ \text{E}[\boldsymbol{x}\boldsymbol{\varepsilon}^{\text{T}}(k)][\text{E}(\boldsymbol{\varepsilon}(k)\boldsymbol{\varepsilon}^{\text{T}}(k))]^{-1}\boldsymbol{\varepsilon}(k)\end{aligned} \tag{3.8}$$

证明： 引入合成向量

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \boldsymbol{\varepsilon}(1) \\ \vdots \\ \boldsymbol{\varepsilon}(k) \end{bmatrix}$$

运用式（3.7）和射影公式，并由 $\text{E}\boldsymbol{\varepsilon}(i) = \boldsymbol{0}$，得到

$$\text{proj}(\boldsymbol{x} \mid \boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k)) = \text{proj}(\boldsymbol{x} \mid \boldsymbol{\varepsilon}(1), \boldsymbol{\varepsilon}(2), \cdots, \boldsymbol{\varepsilon}(k)) =$$

$$\text{proj}(\boldsymbol{x} \mid \boldsymbol{\varepsilon}) = \text{E}\boldsymbol{x} + \text{E}[(\boldsymbol{x} - \text{E}\boldsymbol{x})(\boldsymbol{\varepsilon}^{\text{T}}(1), \boldsymbol{\varepsilon}^{\text{T}}(2), \cdots, \boldsymbol{\varepsilon}^{\text{T}}(k))] \times$$

$$\begin{bmatrix} \text{E}[\boldsymbol{\varepsilon}(1)\boldsymbol{\varepsilon}^{\text{T}}(1)]^{-1} & & 0 \\ & \ddots & \\ 0 & & \text{E}[\boldsymbol{\varepsilon}(1)\boldsymbol{\varepsilon}^{\text{T}}(1)]^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\varepsilon}(1) \\ \vdots \\ \boldsymbol{\varepsilon}(k) \end{bmatrix} =$$

$$\text{E}\boldsymbol{x} + \sum_{i=1}^{k} \text{E}[\boldsymbol{x}\boldsymbol{\varepsilon}^{\text{T}}(i)][\text{E}[\boldsymbol{\varepsilon}(i)\boldsymbol{\varepsilon}^{\text{T}}(i)]^{-1}\boldsymbol{\varepsilon}(i)] =$$

$$\text{E}\boldsymbol{x} + \sum_{i=1}^{k-1} \text{E}[\boldsymbol{x}\boldsymbol{\varepsilon}^{\text{T}}(i)][\text{E}[\boldsymbol{\varepsilon}(i)\boldsymbol{\varepsilon}^{\text{T}}(i)]^{-1}\boldsymbol{\varepsilon}(i)] + \text{E}[\boldsymbol{x}\boldsymbol{\varepsilon}^{\text{T}}(k)][\text{E}[\boldsymbol{\varepsilon}(k)\boldsymbol{\varepsilon}^{\text{T}}(k)]^{-1}\boldsymbol{\varepsilon}(k)] =$$

$$\text{proj}(\boldsymbol{x} \mid \boldsymbol{\varepsilon}(1), \boldsymbol{\varepsilon}(2), \cdots, \boldsymbol{\varepsilon}(k-1)) + \text{E}[\boldsymbol{x}\boldsymbol{\varepsilon}^{\text{T}}(k)][\text{E}[\boldsymbol{\varepsilon}(k)\boldsymbol{\varepsilon}^{\text{T}}(k)]^{-1}\boldsymbol{\varepsilon}(k)] =$$

$$\text{proj}(\boldsymbol{x} \mid \boldsymbol{y}(1), \boldsymbol{y}(2), \cdots, \boldsymbol{y}(k-1)) + \text{E}[\boldsymbol{x}\boldsymbol{\varepsilon}^{\text{T}}(k)][\text{E}[\boldsymbol{\varepsilon}(k)\boldsymbol{\varepsilon}^{\text{T}}(k)]^{-1}\boldsymbol{\varepsilon}(k)]$$

递推射影定理是推导 Kalman 滤波器的递推算法的出发点。

3.1.2　Kalman 滤波器

考虑用如下状态空间模型描述的动态系统

$$\boldsymbol{X}(k+1)=\boldsymbol{\Phi}\boldsymbol{X}(k)+\boldsymbol{\Gamma}\boldsymbol{W}(k) \tag{3.9}$$

$$\boldsymbol{Y}(k)=\boldsymbol{H}\boldsymbol{X}(k)+\boldsymbol{V}(k) \tag{3.10}$$

式中，k 为离散时间，系统在时刻 k 的状态为 $\boldsymbol{X}(k)\in R^n$；$\boldsymbol{Y}(k)\in R^m$ 为对应状态的观测信号；$\boldsymbol{W}(k)\in R^r$ 为输入的白噪声；$\boldsymbol{V}(k)\in R^m$ 为观测噪声。

称式（3.9）为状态方程，称式（3.10）为观测方程。称 $\boldsymbol{\Phi}$ 为状态转移矩阵，$\boldsymbol{\Gamma}$ 为噪声驱动矩阵，$\boldsymbol{H}$ 为观测矩阵。

【假设 1】 $\boldsymbol{W}(k)$ 和 $\boldsymbol{V}(k)$ 是均值为零、方差阵各为 $\boldsymbol{Q}$ 和 $\boldsymbol{R}$ 的不相关白噪声，

$\mathrm{E}\boldsymbol{W}(k)=\boldsymbol{0}$，$\mathrm{E}\boldsymbol{V}(k)=\boldsymbol{0}$，$\mathrm{E}\boldsymbol{W}(k)\boldsymbol{W}^{\mathrm{T}}(j)=\boldsymbol{Q}\delta_{kj}$，$\mathrm{E}\boldsymbol{V}(k)\boldsymbol{V}^{\mathrm{T}}(j)=\boldsymbol{R}\delta_{kj}$，$\boldsymbol{W}(k)$ 和 $\boldsymbol{V}(k)$ 互不相关，因此有 $\mathrm{E}[\boldsymbol{W}(k)\boldsymbol{V}^{\mathrm{T}}(j)]=\boldsymbol{0}$，$\forall k,j$，其中 $\delta_{kk}=1$，$\delta_{kj}=0$，$\forall$ 表示“任意”。

【假设 2】 初始状态 $\boldsymbol{X}(0)$ 不相关于 $\boldsymbol{W}(k)$ 和 $\boldsymbol{V}(k)$，

$$\mathrm{E}[\boldsymbol{X}(0)]=\boldsymbol{\mu}_0,\quad \mathrm{E}[(\boldsymbol{X}(0)-\boldsymbol{\mu}_0)(\boldsymbol{X}(0)-\boldsymbol{\mu}_0)^{\mathrm{T}}]=\boldsymbol{P}_0$$

Kalman 滤波问题是：基于观测信号 $\{\boldsymbol{Y}(1),\boldsymbol{Y}(2),\cdots,\boldsymbol{Y}(k)\}$，求状态 $\boldsymbol{X}(j)$ 的线性最小方差估计值 $\hat{\boldsymbol{X}}(j\,|\,k)$，它极小化性能指标

$$J=\mathrm{E}[(\boldsymbol{X}(j)-\hat{\boldsymbol{X}}(j\,|\,k))^{\mathrm{T}}(\boldsymbol{X}(j)-\hat{\boldsymbol{X}}(j\,|\,k))] \tag{3.11}$$

对于 $j=k$，$j>k$，$j<k$，分别称 $\hat{\boldsymbol{X}}(j\,|\,k)$ 为 Kalman 滤波器、预报器和平滑器。滤波器一般是对当前状态噪声的处理。预报器即为状态预测，通常在导弹拦截、卫星回收等问题上涉及导弹和卫星轨道预测。平滑器主要用在解决卫星入轨初速度估计或卫星轨道重构问题。

在性能指标式（3.11）下，问题归结为求射影

$$\hat{\boldsymbol{X}}(j\,|\,k)=\mathrm{proj}(\boldsymbol{X}(j)\,|\,\boldsymbol{Y}(1),\boldsymbol{Y}(2),\cdots,\boldsymbol{Y}(k)) \tag{3.12}$$

由递推射影【定理 3.1】得到递推关系

$$\hat{\boldsymbol{X}}(k+1\,|\,k+1)=\hat{\boldsymbol{X}}(k+1\,|\,k)+\boldsymbol{K}(k+1)\boldsymbol{\varepsilon}(k+1) \tag{3.13}$$

$$\boldsymbol{K}(k+1)=\mathrm{E}[\boldsymbol{X}(k+1)\boldsymbol{\varepsilon}^{\mathrm{T}}(k+1)]\{\mathrm{E}\boldsymbol{\varepsilon}(k+1)\boldsymbol{\varepsilon}^{\mathrm{T}}(k+1)\}^{-1} \tag{3.14}$$

称 $\boldsymbol{K}(k+1)$ 为 Kalman 滤波器增益。

对状态方程（3.9）两边取射影有

$$\hat{\boldsymbol{X}}(k+1|k)=\boldsymbol{\Phi}\hat{\boldsymbol{X}}(k|k)+\boldsymbol{\Gamma}\text{proj}(\boldsymbol{W}(t)|\boldsymbol{Y}(1),\boldsymbol{Y}(2),\cdots,\boldsymbol{Y}(k)) \tag{3.15}$$

由式（3.9）迭代有

$$\boldsymbol{X}(k)\in L(\boldsymbol{W}(k-1),\cdots,\boldsymbol{W}(0),\boldsymbol{X}(0))$$

且应用式（3.10）有

$$\boldsymbol{Y}(k)\in L(\boldsymbol{V}(k),\boldsymbol{W}(k-1),\cdots,\boldsymbol{W}(0),\boldsymbol{X}(0))$$

因此

$$L(\boldsymbol{Y}(1),\cdots,\boldsymbol{Y}(k))\subset L(\boldsymbol{V}(k),\cdots,\boldsymbol{V}(1),\boldsymbol{W}(k-1),\cdots,\boldsymbol{W}(0),\boldsymbol{X}(0))$$

根据此式、假设 1 和假设 2，有

$$\boldsymbol{W}(k)\perp L(\boldsymbol{Y}(1),\cdots,\boldsymbol{Y}(k))$$

应用射影公式及 $\text{E}\boldsymbol{W}(k)=0$ 可得

$$\text{proj}(\boldsymbol{W}(k)|\boldsymbol{Y}(1),\boldsymbol{Y}(2),\cdots,\boldsymbol{Y}(k))=\boldsymbol{0} \tag{3.16}$$

于是有

$$\hat{\boldsymbol{X}}(k+1|k)=\boldsymbol{\Phi}\hat{\boldsymbol{X}}(k|k) \tag{3.17}$$

同理对观测方程（3.10）两边取射影有

$$\hat{\boldsymbol{Y}}(k+1|k)=\boldsymbol{H}\hat{\boldsymbol{X}}(k+1|k)+\text{proj}(\boldsymbol{V}(k+1)|\boldsymbol{Y}(1),\boldsymbol{Y}(2),\cdots,\boldsymbol{Y}(k)) \tag{3.18}$$

因为 $\boldsymbol{V}(k+1)\perp\boldsymbol{L}(\boldsymbol{Y}(1),\boldsymbol{Y}(2),\cdots,\boldsymbol{Y}(k))$，故有

$$\text{proj}(\boldsymbol{V}(k+1)|\boldsymbol{Y}(1),\boldsymbol{Y}(2),\cdots,\boldsymbol{Y}(k))=\boldsymbol{0}$$

于是有

$$\hat{\boldsymbol{Y}}(k+1|k)=\boldsymbol{H}\hat{\boldsymbol{X}}(k+1|k) \tag{3.19}$$

在这里引出新息的表达式

$$\boldsymbol{\varepsilon}(k+1)=\boldsymbol{Y}(k+1)-\hat{\boldsymbol{Y}}(k+1|k) \tag{3.20}$$

记滤波器和预报估值误差及方差阵为

$$\tilde{\boldsymbol{X}}(k|k)=\boldsymbol{X}(k)-\hat{\boldsymbol{X}}(k|k) \tag{3.21}$$

$$\tilde{\boldsymbol{X}}(k+1|k)=\boldsymbol{X}(k+1)-\hat{\boldsymbol{X}}(k+1|k) \tag{3.22}$$

$$\boldsymbol{P}(k\,|\,k)=\mathrm{E}[\tilde{\boldsymbol{X}}(k\,|\,k)\tilde{\boldsymbol{X}}^{\mathrm{T}}(k\,|\,k)] \tag{3.23}$$

$$\boldsymbol{P}(k+1\,|\,k)=\mathrm{E}[\tilde{\boldsymbol{X}}(k+1\,|\,k)\tilde{\boldsymbol{X}}^{\mathrm{T}}(k+1\,|\,k)] \tag{3.24}$$

则由式（3.10）、式（3.19）和式（3.20）有

$$\boldsymbol{\varepsilon}(t+1)=\boldsymbol{H}\tilde{\boldsymbol{X}}(k+1\,|\,t)+\boldsymbol{V}(k+1) \tag{3.25}$$

由状态方程（3.9）和式（3.19）有

$$\tilde{\boldsymbol{X}}(k+1\,|\,k)=\boldsymbol{\Phi}\tilde{\boldsymbol{X}}(k\,|\,k)+\boldsymbol{\Gamma}\boldsymbol{W}(k) \tag{3.26}$$

由式（3.13）得

$$\tilde{\boldsymbol{X}}(t+1\,|\,k+1)=\tilde{\boldsymbol{X}}(k+1\,|\,k)-\boldsymbol{K}(k+1)\boldsymbol{\varepsilon}(k+1) \tag{3.27}$$

将式（3.25）代入式（3.27）得到

$$\tilde{\boldsymbol{X}}(t+1\,|\,k+1)=[\boldsymbol{I}_n-\boldsymbol{K}(k+1)\boldsymbol{H}]\tilde{\boldsymbol{X}}(k+1\,|\,k)-\boldsymbol{K}(k+1)\boldsymbol{\varepsilon}(k+1) \tag{3.28}$$

式中，$\boldsymbol{I}_n$ 为 $n\times m$ 单位阵。因为

$$\tilde{\boldsymbol{X}}(k\,|\,k)=\boldsymbol{X}(k)-\hat{\boldsymbol{X}}(k\,|\,k)\in L(\boldsymbol{V}(k),\cdots,\boldsymbol{V}(1),\boldsymbol{W}(k-1),\cdots,\boldsymbol{W}(0),\boldsymbol{X}(0))$$

故有 $\boldsymbol{W}(k)\perp\tilde{\boldsymbol{X}}(k\,|\,k)$，则 $\mathrm{E}[\boldsymbol{W}(k)\tilde{\boldsymbol{X}}^{\mathrm{T}}(k\,|\,k)]=\boldsymbol{0}$。

于是由式（3.26）得到

$$\boldsymbol{P}(k+1\,|\,k)=\boldsymbol{\Phi}\boldsymbol{P}(k\,|\,k)\boldsymbol{\Phi}^{\mathrm{T}}+\boldsymbol{\Gamma}\boldsymbol{Q}\boldsymbol{F}^{\mathrm{T}} \tag{3.29}$$

因为

$$\tilde{\boldsymbol{X}}(k+1\,|\,k)=\boldsymbol{X}(k+1)-\hat{\boldsymbol{X}}(k+1\,|\,k)\in L(\boldsymbol{V}(k),\cdots,\boldsymbol{V}(1),\boldsymbol{W}(k),\cdots,\boldsymbol{W}(0),\boldsymbol{X}(0))$$

故有 $\boldsymbol{V}(k+1)\perp\tilde{\boldsymbol{X}}(k+1\,|\,k)$，则 $\mathrm{E}[\boldsymbol{V}(k+1)\tilde{\boldsymbol{X}}^{\mathrm{T}}(k+1\,|\,k)]=\boldsymbol{0}$。

由式（3.25）得到新息方差阵为

$$\mathrm{E}[\boldsymbol{\varepsilon}(k+1)\boldsymbol{\varepsilon}^{\mathrm{T}}(k+1)]=\boldsymbol{H}\boldsymbol{P}(k+1\,|\,k)\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R} \tag{3.30}$$

由式（3.28）可得

$$\begin{aligned}&\boldsymbol{P}(k+1\,|\,k+1)=\mathrm{E}[\boldsymbol{\varepsilon}(k+1)\boldsymbol{\varepsilon}^{\mathrm{T}}(k+1)]\\&=[\boldsymbol{I}_n-\boldsymbol{K}(k+1)\boldsymbol{H}]\boldsymbol{P}(k+1\,|\,k)[\boldsymbol{I}_n-\boldsymbol{K}(t+1)\boldsymbol{H}]^{\mathrm{T}}+\boldsymbol{K}(k+1)\boldsymbol{R}\boldsymbol{H}^{\mathrm{T}}(k+1)\end{aligned} \tag{3.31}$$

下面需要求 Kalman 滤波器的增益 $\boldsymbol{K}(k+1)$，为此先求 $\mathrm{E}[\boldsymbol{X}(k+1)\boldsymbol{\varepsilon}^{\mathrm{T}}(k+1)]$，即

$$\mathrm{E}[\boldsymbol{X}(k+1)\boldsymbol{\varepsilon}^{\mathrm{T}}(k+1)]=\mathrm{E}[(\hat{\boldsymbol{X}}(k+1|k)+\tilde{\boldsymbol{X}}(k+1|k))(\boldsymbol{H}\tilde{\boldsymbol{X}}(k+1|k)+\boldsymbol{V}(k+1))^{\mathrm{T}}] \quad (3.32)$$

因为射影的正交性

$$\hat{\boldsymbol{X}}(k+1|k)\perp\tilde{\boldsymbol{X}}(k+1|k)$$

且注意到$\boldsymbol{V}(k+1)\perp\tilde{\boldsymbol{X}}(k+1|k)$，$\boldsymbol{V}(k+1)\perp\hat{\boldsymbol{X}}(k+1|k)$，于是

$$\mathrm{E}[\boldsymbol{X}(k+1)\boldsymbol{\varepsilon}^{\mathrm{T}}(k+1)]=\boldsymbol{P}(k+1|k)\boldsymbol{H}^{\mathrm{T}} \quad (3.33)$$

将式（3.30）和式（3.33）代入式（3.14），有增益

$$\boldsymbol{P}(k+1|k)=\boldsymbol{\Phi}\boldsymbol{P}(k|k)\boldsymbol{\Phi}^{\mathrm{T}}+\boldsymbol{\Gamma}\boldsymbol{Q}\boldsymbol{\Gamma}^{\mathrm{T}} \quad (3.34)$$

现在用$\boldsymbol{K}(k+1)$的表达式简化$\boldsymbol{P}(k+1|k+1)$的式（3.31），暂时略去式（3.31）右端的时标，将式（3.34）代入式（3.31）有

$$\begin{aligned}\boldsymbol{P}(k+1|k+1)&=[\boldsymbol{I}_n-\boldsymbol{K}\boldsymbol{H}]\boldsymbol{P}-\boldsymbol{P}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}+\boldsymbol{K}\boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}+\boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{\mathrm{T}}\\&=[\boldsymbol{I}_n-\boldsymbol{K}\boldsymbol{H}]\boldsymbol{P}-\boldsymbol{P}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}+\boldsymbol{K}(\boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R})\boldsymbol{K}^{\mathrm{T}}\\&=[\boldsymbol{I}_n-\boldsymbol{K}\boldsymbol{H}]\boldsymbol{P}-\boldsymbol{P}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}+\boldsymbol{P}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}\\&=[\boldsymbol{I}_n-\boldsymbol{K}\boldsymbol{H}]\boldsymbol{P}\end{aligned}$$

即

$$\boldsymbol{P}(k+1|k+1)=[\boldsymbol{I}_n-\boldsymbol{K}(k+1)\boldsymbol{H}]\boldsymbol{P}(k+1|k) \quad (3.35)$$

至此，Kalman 滤波方程组推导完毕，可以将推导结果概况为如下定理。

【定理 3.2】（Kalman 滤波器）系统式（3.9）和式（3.10）在【假设 1】和【假设 2】下，递推 Kalman 滤波器如下。

状态一步预测：$$\hat{\boldsymbol{X}}(k+1|k)=\boldsymbol{\Phi}\hat{\boldsymbol{X}}(k|k) \quad (3.36)$$

状态更新：$$\hat{\boldsymbol{X}}(k+1|k+1)=\hat{\boldsymbol{X}}(k+1|k)+\boldsymbol{K}(k+1)\boldsymbol{\varepsilon}(k+1) \quad (3.37)$$

$$\boldsymbol{\varepsilon}(k+1)=\boldsymbol{Y}(k+1)-\boldsymbol{H}\hat{\boldsymbol{X}}(k+1|k)$$

滤波增益矩阵：$$\boldsymbol{K}(k+1)=\boldsymbol{P}(k+1|k)\boldsymbol{H}^{\mathrm{T}}[\boldsymbol{H}\boldsymbol{P}(k+1|k)\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R}]^{-1} \quad (3.38)$$

一步预测协方差阵：$$\boldsymbol{P}(k+1|k)=\boldsymbol{\Phi}\boldsymbol{P}(k|k)\boldsymbol{\Phi}^{\mathrm{T}}+\boldsymbol{\Gamma}\boldsymbol{Q}\boldsymbol{\Gamma}^{\mathrm{T}} \quad (3.39)$$

协方差阵更新：$$\boldsymbol{P}(k+1|k+1)=[\boldsymbol{I}_n-\boldsymbol{K}(k+1)\boldsymbol{H}]\boldsymbol{P}(k+1|k) \quad (3.40)$$

$$\hat{\boldsymbol{X}}(0|0)=\boldsymbol{\mu}_0,\boldsymbol{P}(0|0)=\boldsymbol{P}_0$$

在一个滤波周期内，从 Kalman 滤波在使用系统信息和观测信息的先后次序来看，Kalman 滤波具有两个明显的信息更新过程：时间更新过程和观测更新过程。式（3.36）说明了根据 k–1 时刻的状态估计预测 k 时刻状态的方法，式（3.39）对这种预测的质量优劣做了定量描述。该两式的计算中仅使用了与系统的动态特性有关的信息，如状态一步转移矩阵、噪声输入阵、过程噪声方差阵。从时间的推移过程来看，该两式将时间从 k–1 时刻推进至 k 时刻，描述了 Kalman 滤波的时间更新过程。其余各式用来计算对时间更新值的修正量，该修正量由时间更新的质量优劣（$\boldsymbol{P}(k|k-1)$）、观测信息的质量优劣（$\boldsymbol{R}$）、观测与状态的关系（$\boldsymbol{H}$）以及具体的观测信息 $\boldsymbol{Y}(k)$ 所确定，所有这些方程围绕一个目的，即正确、合理地利用观测 $\boldsymbol{Y}(k)$，所以这一过程描述了 Kalman 滤波的观测更新过程。

3.1.3　Kalman 滤波的参数处理

1. 噪声矩阵的处理

对于如式（3.9）和式（3.10）描述的系统，$\boldsymbol{W}(k)$和 $\boldsymbol{V}(k)$分别表示过程噪声和测量噪声。一般假设它们为高斯白噪声（White Gaussian Noise），它们的方差分别是 $\boldsymbol{Q}$ 和 $\boldsymbol{R}$（一般假设它们不随系统状态变化而变化）。

在实际应用中，读者会问，如何知道系统的过程噪声 $\boldsymbol{Q}$ 和观测噪声 $\boldsymbol{R}$ 呢？对于观测噪声，也叫测量噪声，它是跟传感器测量精度息息相关的。例如，一个温度计的测量误差是±0.1℃；学生常用刻度尺测量距离，它的误差是±1mm；体重计测量体重的误差是±1g，根据这些信息，我们可以大致知道它们的测量噪声的大小。一般地，观测噪声方差 $\boldsymbol{R}$ 是一个统计意义上的参数，可以理解为：对传感器测量的数据经过长期的概率统计，得出它的测量方差。例如，用温度计测量 100 次房间温度，这 100 次温度数据的方差为 $\hat{\boldsymbol{R}}$，这与该传感器真实方差 $\boldsymbol{R}$ 是非常接近的。同理，对于过程噪声 $\boldsymbol{Q}$，它是过程噪声的方差。例如，在目标跟踪系统中，过程噪声往往是路面摩擦力、空气阻力等因素造成的，在温度测量系统中，过程噪声是由于人体干扰、阳光照射、风等因素造成的，要准确获取 $\boldsymbol{Q}$ 是比较困难的，可以通过对比试验获得。例如，机器人小车在光滑的玻璃上行驶与在粗糙的路面上行驶，两者对比就可以获得在路面上的阻力因素，从而测得阻力噪声方差 $\boldsymbol{Q}$。

2. 特殊情况的处理

已经知道系统（3.9）和（3.10）中的滤波递推方法，对于一些与该系统形式不一致的特殊情况，需要重新讨论怎样把它转化为形同式（3.9）和式（3.10）的最优滤波的问题。

（1）$\boldsymbol{A}$、$\boldsymbol{H}$ 不确定。线性 Kalman 滤波是严格要求系统为线性系统，噪声模型为高斯模型。对于不同系统，它的系统模型 $\boldsymbol{A}$、$\boldsymbol{H}$，状态变量 $\boldsymbol{X}(k)$，噪声 $\boldsymbol{Q}$、$\boldsymbol{R}$ 都是不一样的，要利用 Kalman 滤波处理噪声，首先要建立好系统的数学模型。当考虑到在某些系统中 $\boldsymbol{A}$、$\boldsymbol{H}$ 事先不确定，并且噪声 $\boldsymbol{W}(k)$、$\boldsymbol{V}(k)$ 的统计特性也不知道（即 $\boldsymbol{Q}$、$\boldsymbol{R}$），具体的说是 $\boldsymbol{A}$、$\boldsymbol{H}$ 或者 $\boldsymbol{W}(k)$、$\boldsymbol{V}(k)$ 的统计特性发生变化，首先要估计变化了的参数，进而调整滤波器的增益阵。在这种情况下，一般应用自适应滤波。

（2）含有控制量的系统描述。考虑如下系统，

$$\begin{aligned}\boldsymbol{X}(k)&=\boldsymbol{A}\boldsymbol{X}(k-1)+\boldsymbol{B}\boldsymbol{U}(k-1)+\boldsymbol{\Gamma}\boldsymbol{W}(k-1)\\ \boldsymbol{Y}(k)&=\boldsymbol{H}\boldsymbol{X}(k)+\boldsymbol{V}(k)\end{aligned}\tag{3.41}$$

式中，$\boldsymbol{U}(k)$为控制量。这种情况同式（3.9）和式（3.10）的处理方法是相同的，只需要将控制量 $\boldsymbol{B}\boldsymbol{U}(k)$加到预测式中，增益阵和误差阵的递推式完全一致。

（3）形同式（3.9）和式（3.10），但系统噪声为有色噪声，即有

$$\boldsymbol{W}(k)=\boldsymbol{\Pi}\boldsymbol{W}(k-1)+\boldsymbol{\xi}(k-1)$$

式中，$\boldsymbol{\xi}(k)$ 为白噪声。

处理的办法是将 $\boldsymbol{W}(k)$也列为状态，则增广后的状态为

$$\boldsymbol{X}^{a}(k)=\begin{bmatrix}\boldsymbol{X}(k)\\ \boldsymbol{W}(k)\end{bmatrix}$$

增广后的系统方程和观测方程可写为

$$\begin{bmatrix}\boldsymbol{X}(k)\\ \boldsymbol{W}(k)\end{bmatrix}=\begin{bmatrix}\boldsymbol{A} & \boldsymbol{\Gamma}\\ \boldsymbol{0} & \boldsymbol{\Pi}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}(k-1)\\ \boldsymbol{W}(k-1)\end{bmatrix}+\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{\xi}(k)$$

$$\boldsymbol{Z}(k)=\begin{bmatrix}\boldsymbol{H} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}(k)\\ \boldsymbol{W}(k)\end{bmatrix}+\boldsymbol{V}(k)$$

可以简写为

$$\begin{aligned}\boldsymbol{X}^a(k)&=\boldsymbol{A}^a\boldsymbol{X}^a(k-1)+\boldsymbol{\Gamma}^a\boldsymbol{W}(k-1)\\ \boldsymbol{Z}^a(k)&=\boldsymbol{H}^a\boldsymbol{X}^a(k)+\boldsymbol{V}(k)\end{aligned}\tag{3.42}$$

此时即符合式（3.9）和式（3.10）所描述的一般形式。

Kalman 滤波算法具有如下特点。

（1）由于 Kalman 滤波算法将被估计的信号看作在白噪声作用下一个随机线性系统的输出，并且其输入/输出关系是由状态方程和输出方程在时间域内给出的，因此这种滤波方法不仅适用于平稳随机过程的滤波，而且特别适用于非平稳或平稳马尔可夫序列或高斯-马尔可夫序列的滤波，所以其应用范围是十分广泛的。

（2）Kalman 滤波算法是一种时间域滤波方法，采用状态空间描述系统。系统的过程噪声和量测噪声并不是需要滤除的对象，它们的统计特性正是估计过程中需要利用的信息，而被估计量和观测量在不同时刻的一、二阶矩却是不必要知道的。

（3）由于 Kalman 滤波的基本方程是时间域内的递推形式，其计算过程是一个不断地“预测-修正”的过程，在求解时不要求存储大量数据，并且一旦观测到了新的数据，随即可以算得新的滤波值，因此这种滤波方法非常适合于实时处理、计算机实现。

（4）由于滤波器的增益矩阵与观测无关，因此它可预先离线算出，从而可以减少实时在线计算量。在求滤波器增益矩阵时，要求一个矩阵的逆，它的阶数只取决于观测方程的维数，而该维数通常很小，这样，求逆运算是比较方便的。另外，在求解滤波器增益的过程中，随时可以算得滤波器的精度指标 P，其对角线上的元素就是滤波误差向量各分量的方差。

3.2　Kalman 滤波在温度测量中的应用

3.2.1　原理介绍

到这里，读者也许还没弄明白 Kalman 滤波到底是怎么样的一个过程。在此结合温度测量的例子，使读者可以直观了解 Kalman 滤波。

假设我们要研究的对象是一个房间的温度。根据经验判断，这个房间的温度大概在 25℃左右，可能受空气流通、阳光等因素影响，房间内温度会小幅度地波动。我们以分钟为单位，定时测量房间温度，这里的 1 分钟，可以理解为采样时

间。假设测量温度时，外界的天气是多云，阳光照射时有时无，同时房间不是 100%密封的，可能有微小的与外界空气的交换，即引入过程噪声 $\boldsymbol{W}(k)$，其方差为 $\boldsymbol{Q}$，大小假定为 $\boldsymbol{Q}$=0.01（假如不考虑过程噪声的影响，即真实温度是恒定的，那么这时候 $\boldsymbol{Q}$=0）。对照式（3.9）和式（3.10），相应地，$\boldsymbol{A}$=1，$\boldsymbol{\Gamma}$=1，$\boldsymbol{Q}$=0.01，状态 $\boldsymbol{X}(k)$是在第 k 分钟时的房间温度，是一维的。那么该系统的状态方程可以写为

$$\boldsymbol{X}(k)=\boldsymbol{X}(k-1)+\boldsymbol{W}(k)$$

现在用温度计开始测量房间的温度，假设温度计的测量误差为±0.5℃，从出厂说明书上我们得知该温度计的方差为 0.25。也就是说，温度计第 k 次测量的数据不是 100%准确的，它是有测量噪声 $\boldsymbol{V}(k)$的，并且其方差 $\boldsymbol{R}$=0.25，因此测量方程为 $\boldsymbol{Z}(k)=\boldsymbol{X}(k)+\boldsymbol{V}(k)$。

到此，读者很容易想到，该系统的状态和观测方程为

$$\begin{aligned}\boldsymbol{X}(k)&=\boldsymbol{A}\boldsymbol{X}(k-1)+\boldsymbol{\Gamma}\boldsymbol{W}(k-1)\\ \boldsymbol{Z}(k)&=\boldsymbol{H}\boldsymbol{X}(k)+\boldsymbol{V}(k)\end{aligned}\tag{3.43}$$

式中，$\boldsymbol{X}(k)$是一维变量温度；$\boldsymbol{A}$=1；$\boldsymbol{\Gamma}$=1；$\boldsymbol{H}$=1；$\boldsymbol{W}$(k)和 $\boldsymbol{V}$(k)的方差为 $\boldsymbol{Q}$ 和 $\boldsymbol{R}$。

模型建好以后，就可以利用 Kalman 滤波了。假如要估算第 k 时刻的实际温度值，首先要根据第 k–1 时刻的温度值来预测 k 时刻的温度。

（1）假定第 k–1 时刻的温度值测量值为 23.9℃，房间真实温度为 24.0℃，该测量值的偏差是 0.1℃，即协方差 $\boldsymbol{P}(k-1)=0.1^2$。

（2）在第 k 时刻，房间的真实温度是 24.1℃，温度计在该时刻测量的值为 24.5℃，偏差为 0.4℃。我们用于估算第 k 时刻的温度有两个温度值，分别是 k–1 时刻 23.9℃和 k 时刻的 24.5℃，如何融合这两组数据，得到最逼近真实值的估计呢？

首先，利用 k–1 时刻温度值预测第 k 时刻的温度，其预计偏差为 $P(k|k-1)=\boldsymbol{P}(k-1)+\boldsymbol{Q}$=0.02，计算 Kalman 增益 $\boldsymbol{K}=P(k|k-1)/(P(k|k-1)+\boldsymbol{R})$=0.0741，那么这时候利用 k 时刻的观测值，得到温度的估计值为 $\boldsymbol{X}(k)$=23.9+0.0741×(24.1–23.9)=23.915℃。可见，与 23.9℃和 24.5℃相比较，Kalman 估计值 23.915℃更接近真实值 24.1℃。此时更新 k 时刻的偏差 $\boldsymbol{P}(k)=(1-\boldsymbol{K}*\boldsymbol{H})\ P(k|k-1)$=0.0186。最后由 $\boldsymbol{X}(k)$=23.915℃和 $\boldsymbol{P}(k)$= 0.0186，可以继续对下一时刻观测数据 $\boldsymbol{Z}(k+1)$进行更新和处理。

（3）这样，Kalman 滤波器就不断地把方差递归，从而估算出最优的温度值。

当然，我们需要确定 Kalman 滤波器两个初始值，分别是 $\boldsymbol{X}(0)$和 $\boldsymbol{P}(0)$。

运行 3.2.2 节所附程序得到如图 3.3 和图 3.4 所示的仿真结果。

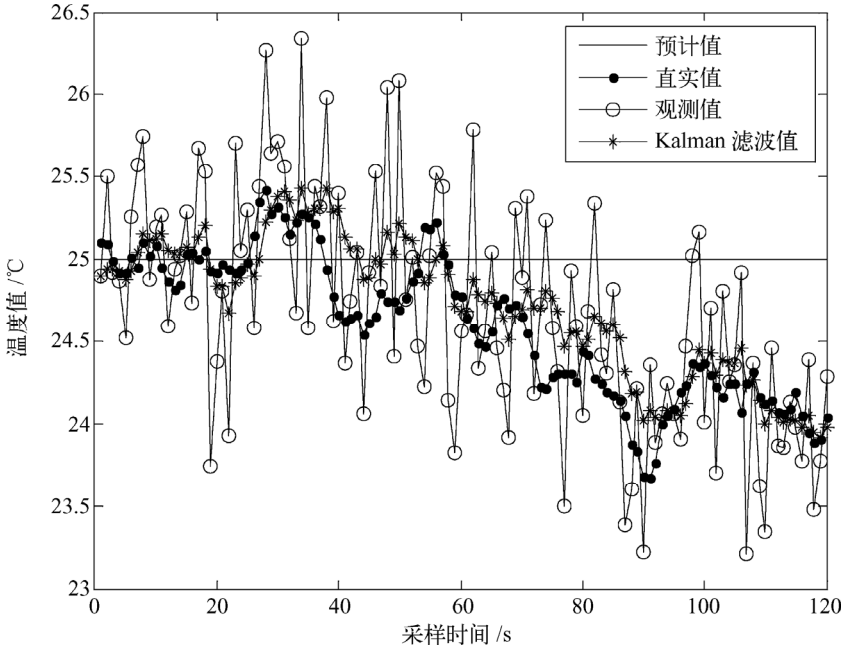


图 3.3　房间温度估计

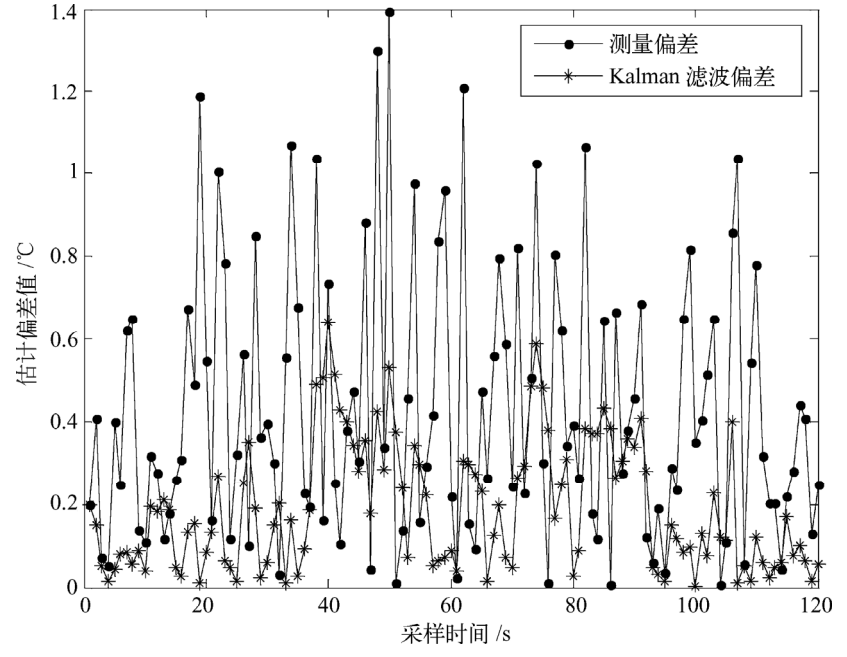


图 3.4　房间温度误差分析

从图上可以看出，Kalman 滤波与温度计直接测量的值相比，大大降低了偏差，虽然 Kalman 滤波误差没有完全消失，但它使状态尽可能地逼近真实值。

3.2.2 MATLAB 仿真程序

```
%%%
```

```
    % 温度计对 k 时刻房间温度的测量，Kalman 滤波是站在温度计角度进行的，
    % 它不知道此刻真实状态 X(k)，只能利用本次测量值 Z(k)和上一次估计值
    % Xkf(k)来做处理，其目标是最大限度地降低测量噪声 R 的影响，尽可能
    % 地逼近 X(k)，这也是 Kalman 滤波目的所在
    Z(k)=H*X(k)+V(k);
    % 第三步：Kalman 滤波
    % 有了 k 时刻的观测 Z(k)和 k-1 时刻的状态，那么就可以进行滤波了，
    % 读者可以对照式（3.36）到式（3.40），理解滤波过程
    X_pre=F*Xkf(k-1);                % 状态预测
    P_pre=F*P(k-1)*F'+Q;             % 协方差预测
    Kg=P_pre*inv(H*P_pre*H'+R);      % 计算 Kalman 增益
    e=Z(k)-H*X_pre;                  % 新息
    Xkf(k)=X_pre+Kg*e;               % 状态更新
    P(k)=(I-Kg*H)*P_pre;             % 协方差更新
end
%%%
```

3.3 Kalman 滤波在自由落体运动目标跟踪中的应用

3.3.1 状态方程的建立

考察如图 3.5 所示的问题。某一物体在重力场做自由落体运动，观测装置对其位移进行检测，在传感器受到未知的独立分布随机信号的干扰下，我们需要估计该物体的运动位移和速度。

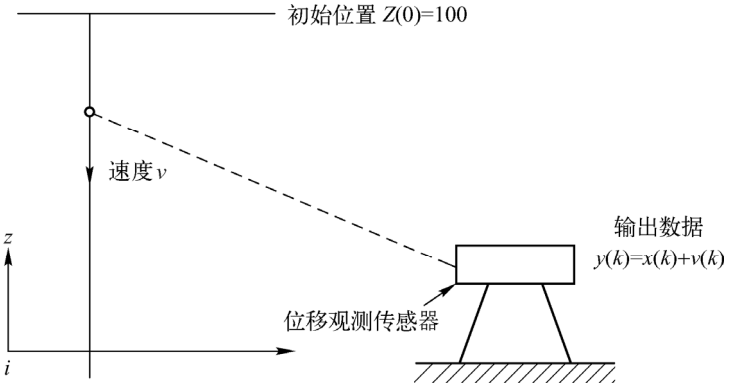


图 3.5　自由落体观测系统

对这样一个无噪声的二阶系统，它处于一个保守场中，即

$$\ddot{z} = -g, t \geqslant 0 \tag{3.44}$$

设物体的位移 $z = x_1$ 和速度 $\ddot{z} = x_2$，定义如下的向量：

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.45}$$

现在推导该自由落体目标的状态转移矩阵。由已有的运动学方程，容易得出该运动物体的状态方程

$$\boldsymbol{x}(k) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}(k-1) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} (-g) \tag{3.46}$$

给定位置观测装置，在测量值受到某种独立，随机干扰 $v(k)$ 的影响时，其观测方程可写为

$$y(k) = x(k) + v(k) \tag{3.47}$$

即

$$y(k)=[1\quad 0]\boldsymbol{x}(k)+v(k) \tag{3.48}$$

给定 $v(k)$ 的方差 $R(k)=\sigma_\sigma^2=1$，物体的初始状态 $\boldsymbol{x}(t)=\begin{bmatrix}95\\1\end{bmatrix}$，初始误差为 $\boldsymbol{P}(0)=\begin{bmatrix}\sqrt{100} & 0\\0 & \sqrt{1}\end{bmatrix}$。

且由运动方程的物理模型可知

$$\boldsymbol{Q}(0)=\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}=\boldsymbol{0}$$

定义均方误差

$$\boldsymbol{P}(k)=\mathrm{E}[\boldsymbol{e}^2(k)] \tag{3.49}$$

将式（3.45）代入式（3.49）并求导可得出 Kalman 滤波的递推过程。估计器

$$\hat{\boldsymbol{x}}(k)=\boldsymbol{A}\hat{\boldsymbol{x}}(k-1)+\boldsymbol{K}(k)[y(k)-\boldsymbol{CA}\hat{\boldsymbol{x}}(k-1)] \tag{3.50}$$

其中的递推关系式为

$$\begin{aligned}&\boldsymbol{P}_1(k)=\boldsymbol{A}\boldsymbol{P}(k-1)\boldsymbol{A}^{\mathrm{T}}+\boldsymbol{Q}(k-1)\\&\boldsymbol{K}(k)=\boldsymbol{P}_1(k)\boldsymbol{C}^{\mathrm{T}}[\boldsymbol{C}\boldsymbol{P}_1(k)\boldsymbol{C}^{\mathrm{T}}+R]^{-1}\\&\boldsymbol{P}(k)=\boldsymbol{P}_1(k)-\boldsymbol{K}(k)\boldsymbol{C}\boldsymbol{P}_1(k)\end{aligned} \tag{3.51}$$

考虑估计器表达式（3.50），把它分成两个部分，前者为预测，后者为修正：

$$\hat{\boldsymbol{x}}(k)=\underbrace{\boldsymbol{A}\hat{\boldsymbol{x}}(k-1)}_{\text{预测项}}+\underbrace{\underbrace{\boldsymbol{K}(k)}_{\text{增益矩阵}}\ [y(k)-\boldsymbol{CA}\hat{\boldsymbol{x}}(k-1)]}_{\text{修正项}} \tag{3.52}$$

第 k 时刻的估计是由第 k–1 时刻的预测值加上修正量得到的，那么很容易得到 k+1 时刻的预测值：

$$\hat{\boldsymbol{x}}(k+1\,|\,k)=\boldsymbol{A}\hat{\boldsymbol{x}}(k) \tag{3.53}$$

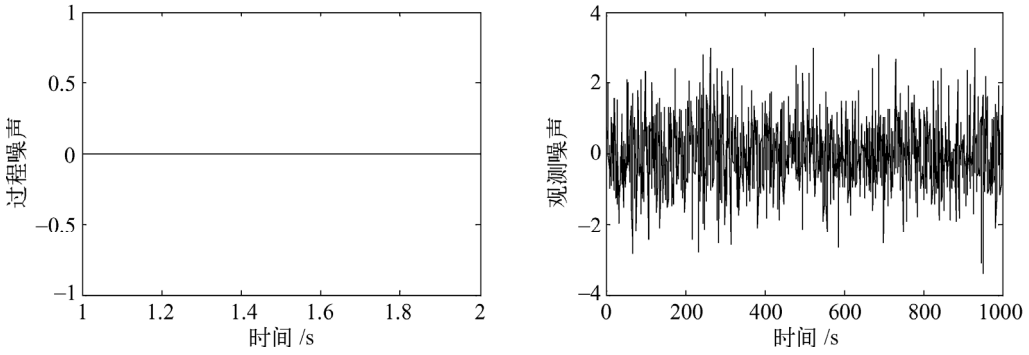
将式（3.53）结合估计器的表达式，则得到 Kalman 的预测过程。Kalman 预测器为

$$\hat{\boldsymbol{x}}(k+1\,|\,k)=\boldsymbol{A}\hat{\boldsymbol{x}}(k)=\boldsymbol{A}\{\boldsymbol{A}\hat{\boldsymbol{x}}(k-1)+\boldsymbol{K}(k)[y(k)-\boldsymbol{CA}\hat{\boldsymbol{x}}(k-1)]\} \tag{3.54}$$

其中的递推关系为式（3.51）。

按上述算法流程，对如式（3.46）描述的自由落体运动目标进行 Kalman 滤波

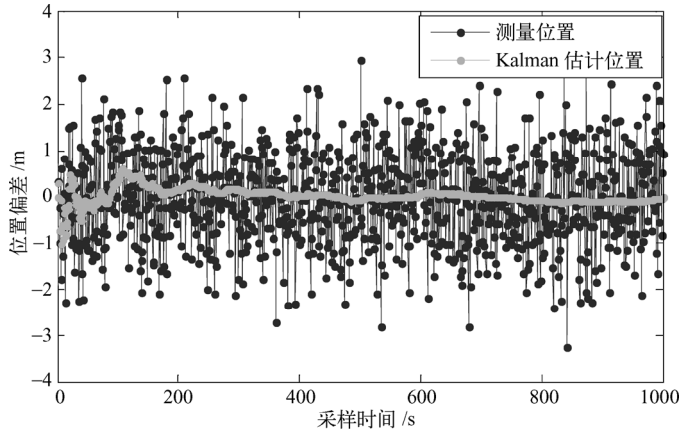
和预测。图 3.6 是系统的噪声，在本例中忽略空气噪声等的影响，即过程噪声为 Q=0，图 3.6（a）展示了这一结果，当然读者也可以将 Q 设为非零。图 3.6（b）展示了均值为 0、方差为 1 的测量噪声，可以看出，噪声最大值逼近 4，这是非常大的误差，意味着单次测量位置的偏差最大可达到 4m，这样的测距传感器应该早被淘汰了。



（a）过程噪声　　（b）测量噪声

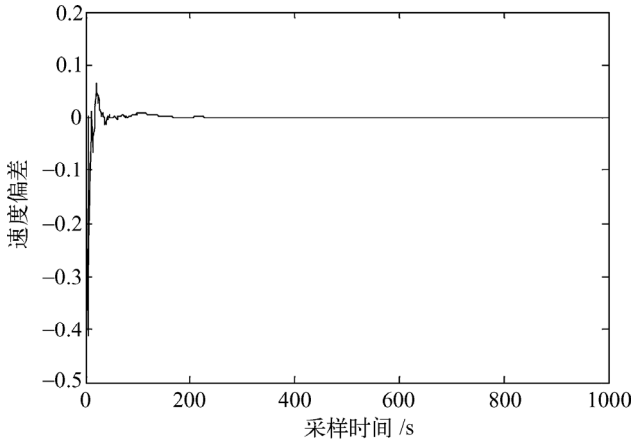
图 3.6　系统的噪声

图 3.7 是 Kalman 滤波算法对噪声的处理。从图 3.7（a）位置估计来看，测量值受到测量噪声的污染，但是 Kalman 滤波算法则很好地降低了噪声的干扰，在 k=200s 以后，位置偏差接近 0。图 3.7（b）是 Kalman 递推算法得到的速度偏差，容易看出，在经过少数的几次迭代后误差很快得到收敛。



（a）观测位置、滤波位置与真实值之间的偏差

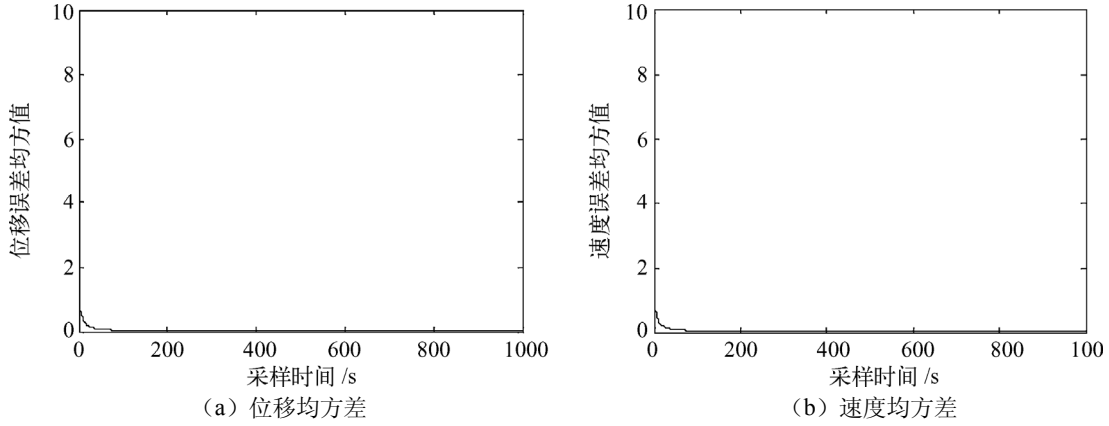
图 3.7　Kalman 滤波算法对噪声的处理



（b）Kalman 滤波后的速度与真实值之间的偏差

图 3.7　Kalman 滤波算法对噪声的处理（续）

图 3.8 是 Kalman 滤波算法在各个时刻状态的协方差，图 3.8（a）是位移的误差均方值，图 3.8（b）是速度的误差均方值，它们都有较好的收敛性，可见线性 Kalman 滤波对高斯噪声的处理是非常有效的。



（a）位移均方差　　（b）速度均方差

图 3.8　Kalman 滤波算法在各个时刻状态的协方差

3.3.2　MATLAB 仿真程序

```
%%%
```

```
Q=[0,0;0,0]; % 过程噪声方差为 0，即下落过程忽略空气阻力
R=1;              % 观测噪声方差
W=sqrt(Q)*randn(2,N);% 既然 Q 为 0，则 W=0；在此写出，方便对照理解
V=sqrt(R)*randn(1,N);% 测量噪声 V(k)
% 系统矩阵
A=[1,1;0,1]; %状态转移矩阵
B=[0.5;1];    %控制量
U=-1;
H=[1,0];       %观测矩阵
% 初始化
X=zeros(2,N);   % 物体真实状态
X(:,1)=[95;1]; % 初始位移和速度
P0=[10,0;0,1]; % 初始误差
Z=zeros(1,N);
Z(1)=H*X(:,1); % 初始观测值
Xkf=zeros(2,N);% Kalman 估计状态初始化
Xkf(:,1)=X(:,1);
err_P=zeros(N,2);
err_P(1,1)=P0(1,1);
err_P(1,2)=P0(2,2);
I=eye(2);          % 二维系统
%%%
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 误差计算
messure_err_x=zeros(1,N); % 位移的测量误差
kalman_err_x=zeros(1,N);% Kalman 估计的位移与真实位移之间的偏差
kalman_err_v=zeros(1,N);% Kalman 估计的速度与真实速度之间的偏差
for k=1:N
    messure_err_x(k)=Z(k)-X(1,k);
    kalman_err_x(k)=Xkf(1,k)-X(1,k);
    kalman_err_v(k)=Xkf(2,k)-X(2,k);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 画图输出
% 噪声图
figure
plot(W);
xlabel('采样时间/s');
ylabel('过程噪声');
figure
plot(V);
xlabel('采样时间/s');
ylabel('测量噪声');
% 位置偏差
figure
hold on,box on;
plot(messure_err_x,'-r.'); %测量的位移误差
plot(kalman_err_x,'-g.'); %kalman 估计位置误差
legend('测量位置','Kalman 估计位置')
xlabel('采样时间/s');
ylabel('位置偏差/m');
%Kalman 速度偏差
figure
plot(kalman_err_v);
xlabel('采样时间/s');
ylabel('速度偏差');
% 均方值
figure
plot(err_P(:,1));
xlabel('采样时间/s');
```

```
ylabel('位移误差均方值');
figure
plot(err_P(:,1));
xlabel('采样时间/s');
ylabel('速度误差均方值');
%%%
```

表示在时刻 kT_0 处 GPS 定位的观测值，则有观测模型

$$y(k)=s(k)+v(k) \tag{3.55}$$

式中，$v(k)$表示 GPS 定位误差（观测噪声），可假设它是零均值、方差为 σ_v^2 的白噪声，方差 σ_v^2 可以通过大量 GPS 观测试验数据用统计方法获取。记在时刻 kT_0 处船舶速度为 $\dot{s}(k)$，加速度为 $a(k)$，由匀加速运动公式有

$$s(k+1)=s(k)+\dot{s}(k)T_0+0.5T_0^2a(k) \tag{3.56}$$

$$\dot{s}(k+1)=\dot{s}(k)+T_0a(k) \tag{3.57}$$

而加速度 $a(k)$由机动加速度 $u(k)$和随机加速度 $w(k)$两部分合成，即

$$a(k)=u(k)+w(k) \tag{3.58}$$

式中，$u(k)$为船舶动力系统的控制信号，它是人为输出的已知机动信号；$w(k)$是由海风和海浪引起的随机加速度，假设它是零均值、方差为 σ_w^2 的独立于 $v(k)$的白噪声。定义在采样时刻 kT_0 处系统的状态 $x(k)$为船舶的位置和速度，即

$$x(k)=\begin{bmatrix}s(k)\\ \dot{s}(k)\end{bmatrix} \tag{3.59}$$

可得到船舶运动的状态方程

$$\begin{bmatrix}s(k+1)\\ \dot{s}(k+1)\end{bmatrix}=\begin{bmatrix}1 & T_0\\ 0 & 1\end{bmatrix}\begin{bmatrix}s(k)\\ \dot{s}(k)\end{bmatrix}+\begin{bmatrix}0.5T_0^2\\ T_0\end{bmatrix}u(k)+\begin{bmatrix}0.5T_0^2\\ T_0\end{bmatrix}w(k) \tag{3.60}$$

观测方程为

$$y(k)=[1\quad 0]\begin{bmatrix}s(k)\\ \dot{s}(k)\end{bmatrix}+v(k) \tag{3.61}$$

即系统的状态空间模型为

$$\begin{aligned}\boldsymbol{x}(k+1)&=\boldsymbol{\Phi x}(k)+\boldsymbol{B}u(k)+\boldsymbol{\Gamma}w(k)\\ y(k)&=\boldsymbol{Hx}(k)+v(k)\end{aligned} \tag{3.62}$$

式中，

$$\boldsymbol{\Phi}=\begin{bmatrix}1 & T_0\\ 0 & 1\end{bmatrix},\ \boldsymbol{B}=\boldsymbol{\Gamma}=\begin{bmatrix}0.5T_0^2\\ T_0\end{bmatrix},\ \boldsymbol{H}=[1\quad 0]$$

于是船舶 GPS 导航定位 Kalman 滤波问题是：基于 GPS 观测数据（$y(1),y(2),\cdots,y(k)$），得到船舶在 k 时刻的位置 $s(k)$的最优估计 $\hat{s}(k\,|\,k)$。

在不考虑机动目标自身的动力因素时（$u(k)$=0），将匀速直线运动的船舶系统推广到四维，即

$$\boldsymbol{X}(k)=[x(k)\ \dot{x}(k)\ y(k)\ \dot{y}(k)]^{\mathrm{T}} \tag{3.63}$$

状态包含水平方向的位置和速度和纵向的位置和速度，则系统方程可以用下式表示。

$$\begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \end{bmatrix}=\begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x(k-1) \\ \dot{x}(k-1) \\ y(k-1) \\ \dot{y}(k-1) \end{bmatrix}+\begin{bmatrix} 0.5T^2 & 0 \\ T & 0 \\ 0 & 0.5T^2 \\ 0 & T \end{bmatrix}\boldsymbol{w}_{2\times 1}(k) \tag{3.64}$$

$$\boldsymbol{Z}(k)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \end{bmatrix}+\boldsymbol{v}_{2\times 1}(k) \tag{3.65}$$

假定船舶在二维水平面上运动，初始位置为（−100m, 200m），水平运动速度为 2m/s，垂直方向的运动速度为 20 m/s，GPS 接收机的扫描周期为 T=1s，观测噪声的均值为 0，方差为 100。过程噪声越小，目标越接近匀速直线运动；反之，则为曲线运动。仿真得到以下结果。

图 3.10 中观测轨迹明显在振荡，说明测量噪声影响非常大，而经过 Kalman 滤波后，滤波估计比较接近目标的真实运动轨迹。

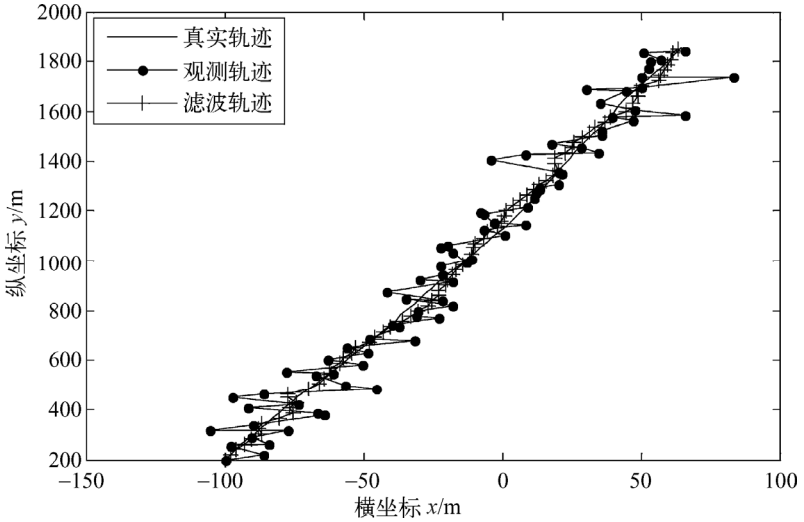


图 3.10　跟踪轨迹图

图 3.11 可以看出位移的观测噪声最大值接近 35m，对于目标运动场地（长约 1800m，宽约 250m）来说，这个噪声非常大，当然这也只限于仿真，实际传感器的测量误差不可能这么大。经过 Kalman 滤波之后，位置偏差降低到 10m 以下，可以看出，Kalman 滤波虽然不能完全消除噪声，但是它已经是最大限度地降低噪声的影响了。

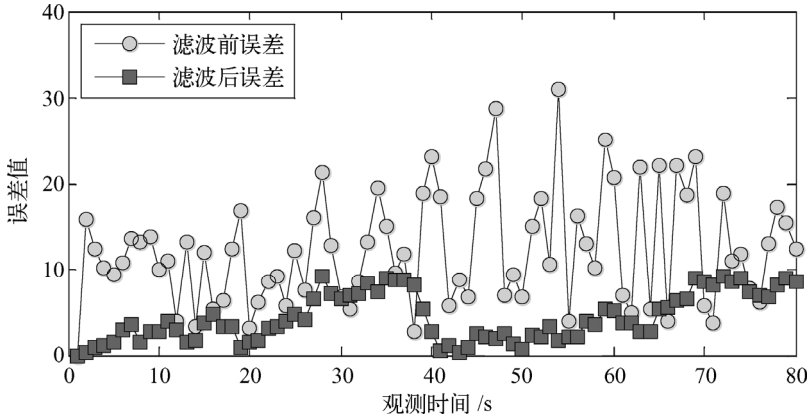


图 3.11　跟踪误差图

3.4.2　MATLAB 仿真程序

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 功能说明：Kalman 滤波在船舶 GPS 导航定位系统中的应用
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function main
clc;clear;
T=1;%雷达扫描周期
N=80/T; %总的采样次数
X=zeros(4,N); % 目标真实位置、速度
X(:,1)=[-100,2,200,20];% 目标初始位置、速度
Z=zeros(2,N); % 传感器对位置的观测
Z(:,1)=[X(1,1),X(3,1)];  % 观测初始化
delta_w=1e-2;  %如果增大这个参数，目标真实轨迹就是曲线了
Q=delta_w*diag([0.5,1,0.5,1]) ; % 过程噪声均值
R=100*eye(2);  %观测噪声均值
F=[1,T,0,0;0,1,0,0;0,0,1,T;0,0,0,1];  % 状态转移矩阵
H=[1,0,0,0;0,0,1,0];    % 观测矩阵
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for t=2:N
```

```
        X(:,t)=F*X(:,t-1)+sqrtm(Q)*randn(4,1);%目标真实轨迹
        Z(:,t)=H*X(:,t)+sqrtm(R)*randn(2,1); %对目标观测
    end
    % Kalman 滤波
    Xkf=zeros(4,N);
    Xkf(:,1)=X(:,1); % Kalman 滤波状态初始化
    P0=eye(4); %  协方差阵初始化
    for i=2:N
        Xn=F*Xkf(:,i-1); %预测
        P1=F*P0*F'+Q;%预测误差协方差
        K=P1*H'*inv(H*P1*H'+R);%增益
        Xkf(:,i)=Xn+K*(Z(:,i)-H*Xn);%状态更新
        P0=(eye(4)-K*H)*P1;%滤波误差协方差更新
    end
    %  误差分析
    for i=1:N
        Err_Observation(i)=RMS(X(:,i),Z(:,i)); %  滤波前的误差
        Err_KalmanFilter(i)=RMS(X(:,i),Xkf(:,i)); %  滤波后的误差
    end
    %%%%
```

```
if length(X2)<=2
    dist=sqrt( (X1(1)-X2(1))^2 + (X1(3)-X2(2))^2 );
else
    dist=sqrt( (X1(1)-X2(1))^2 + (X1(3)-X2(3))^2 );
end
%%%
```

含是否有油田和油田几何形状大小的重要信息，通常可用 Bernoulli-Gaussian 白噪声描述，即

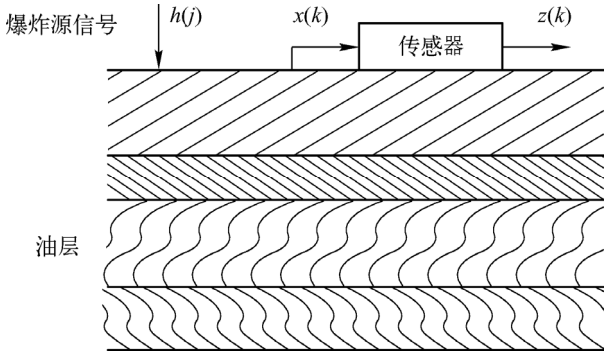


图 3.13　石油地震勘探原理

$$w(k)=b(k)g(k) \tag{3.68}$$

式中，$b(k)$ 为取值 0 和 1 的 Bernoulli 白噪声，取值概率为

$$P(b(k))=\begin{cases}\lambda, & b(k)=1\\ 1-\lambda, & b(k)=0\end{cases} \tag{3.69}$$

而 $g(k)$ 是均值为 0、方差为 σ_g^2，且独立于 $b(k)$ 的 Gaussian 白噪声。石油地震勘探的关键技术在于寻求 $w(k)$ 的最优估计值。由式（3.66）和式（3.67）看到，反射系数序列 $w(k)$ 是卷积模型的输入。由卷积模型的输出 $z(k)$ 估计其输入 $w(k)$ 叫反卷积。

由式（3.68）和式（3.69）引出 $w(k)$ 取非零值时的概率 λ，显然所引出的 $w(t)$ 是均值为零、方差为 $\sigma_w^2=\lambda\sigma_g^2$ 的白噪声。

卷积模型（3.66）和（3.67）可以表示为如下状态空间模型：

$$\boldsymbol{x}(k+1)=\boldsymbol{\Phi}\boldsymbol{x}(k)+\boldsymbol{\Gamma}w(k) \tag{3.70}$$

$$\boldsymbol{z}(k)=\boldsymbol{H}\boldsymbol{x}(k)+v(k) \tag{3.71}$$

式中，$\boldsymbol{x}(k)\in R^n$；$\boldsymbol{\Phi}$、$\boldsymbol{\Gamma}$ 和 $\boldsymbol{H}$ 分别为 $n\times n$、$n\times 1$ 和 $1\times n$ 的矩阵。初值 $\boldsymbol{x}(0)=\boldsymbol{0}$，事实上，由式（3.70）迭代有

$$z(k)=\boldsymbol{H}\boldsymbol{\Phi}^k\boldsymbol{x}(0)+\sum_{j=1}^{k}\boldsymbol{H}\boldsymbol{\Phi}^{k-j}\boldsymbol{\Gamma}w(j)+v(k) \tag{3.72}$$

由假设 $\boldsymbol{x}(0)=\boldsymbol{0}$ 有

$$z(k)=\sum_{j=1}^{k}\boldsymbol{H}\boldsymbol{\Phi}^{k-j}\boldsymbol{\Gamma}w(j)+v(k) \tag{3.73}$$

比较式（3.72）和式（3.73），有

$$h(i)=\boldsymbol{H}\boldsymbol{\Phi}^{i}\boldsymbol{\Gamma}\text{，}\ i=1,2,3,\cdots \tag{3.74}$$

系数 $\boldsymbol{H}\boldsymbol{\Phi}^{i}\boldsymbol{\Gamma}$ 被称为 Markov 参数。由已知的脉冲响应序列 $\{h(0),h(1),h(2),\cdots\}$ 求矩阵 $\boldsymbol{\Phi}$、$\boldsymbol{\Gamma}$ 和 $\boldsymbol{H}$ 及状态维数 n 的问题叫实现问题。

综上所述，石油地震勘探问题可归结为在状态空间模型（3.66）和（3.67）下基于输出 $z(k)$ 估计其输入 $w(k)$ 的反卷积估值器。

3.5.2　石油地震勘探白噪声反卷积滤波仿真实现

石油地震勘探输出白噪声（反射系数序列）最优估计问题如图 3.14 所示，下面给出一个仿真例子说明白噪声反卷积滤波器的性能。

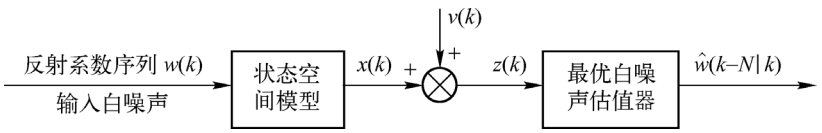


图 3.14　石油地震勘探输出白噪声最优估计问题

考虑随机系统

$$\boldsymbol{x}(k+1)=\begin{bmatrix}1 & 0\\ 0.3 & -0.5\end{bmatrix}\boldsymbol{x}(k)+\begin{bmatrix}-1\\ 2\end{bmatrix}w(k) \tag{3.75}$$

$$z(k)=\begin{bmatrix}1 & 1\end{bmatrix}\boldsymbol{x}(k)+v(k) \tag{3.76}$$

式中，$w(k)=b(k)g(k)$ 是 Bernoulli-Gaussian 白噪声；$b(k)$ 是取值为 0 和 1 的 Bernoulli 白噪声，概率取值如式（3.69）；$g(k)$ 是均值为零、方差为 σ_g^2 且独立于 $b(k)$ 的高斯白噪声。易知 $\mathrm{E}[b(k)]=\lambda$，$\mathrm{E}[b^2(k)]=\lambda$，$\sigma_w^2=\mathrm{E}[w^2(k)]=\mathrm{E}[b^2(k)]=\mathrm{E}[g^2(k)]=\lambda\sigma_g^2$。

仿真过程中，取 $\lambda=0.3$，$\sigma_v^2=0.1$，$\sigma_g^2=49$，仿真结果如图 3.15 所示。图中，实线端点纵坐标为 $w(k)$，代表真实值；圆点纵坐标表示 $w(k)$的 Kalman 滤波的估计值。可以看到，$w(k)$是取值非零且稀疏的白噪声，$w(k)$只在个别处不为 0，且 $w(k)$取非零值的幅度和出现非零值的时刻都是随机的。仿真结果表明，$\hat{w}(k\,|\,k+2)$ 的估计精度比 $\hat{w}(k\,|\,k+1)$ 高，而 $\hat{w}(k\,|\,k+3)$ 又比 $\hat{w}(k\,|\,k+2)$ 的估计精度高。

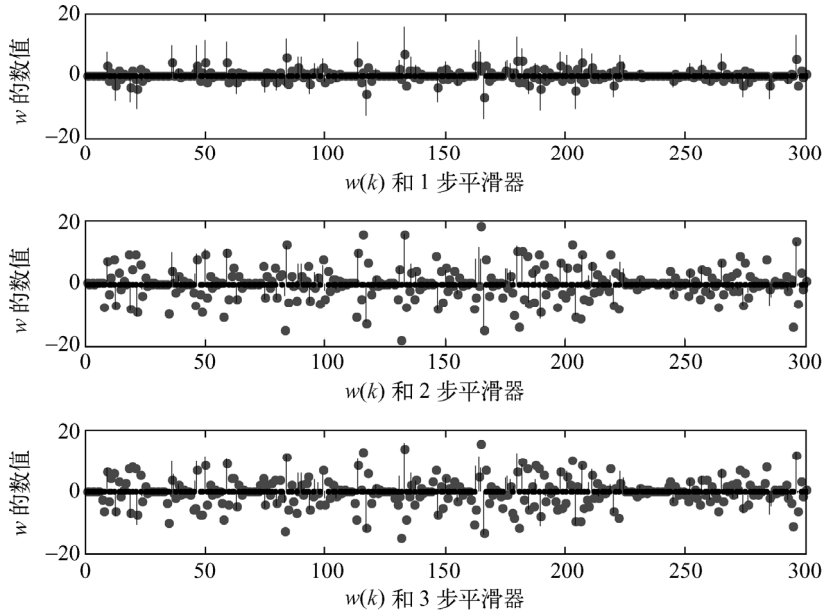


图 3.15　石油地震勘探输出白噪声估值器

3.5.3　MATLAB 仿真程序

```
%%%
```

```
rand('state',1);
para=rand(1,T+10);   % 产生 0-1 之间高斯分布的随机数值
for t=1:T+10
    if para(t)<longa
        b(t)=1;
    else
        b(t)=0;
    end
    w(t)=b(t)*g(t);   % 产生 w(t)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 状态空间模拟部分
% 观测噪声 v(t)产生
v=sqrt(R)*randn(1,T+10);
% 产生状态和观测信息
X=zeros(2,T+10);
Z=zeros(1,T+10);
Z(1)=H*X(:,1)+v(1);
for t=2:T+10
    X(:,t)=F*X(:,t-1)+L*w(t-1);   % 状态方程
    Z(t)=H*X(:,t)+v(t); % 观测方程
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kalman 滤波部分
P0=eye(n);
Xe=zeros(n,T+10);
PP=[];
for t=1:T+8
    XX=F*X(:,t); % 状态预测
    % 计算协方差矩阵 P
    P=F*P0*F'+L*Q*L';
    PP=[PP,P];
    % 计算 Kalman 增益
    K(:,t)=P*H'*inv(H*P*H'+R);
    % 计算新息
    e(:,t)=Z(t)-H*XX;
    % 状态更新
    Xe(:,t)=XX+K(:,t)*e(:,t);
    % 方差更新
```

```
        P0=(eye(n)-K(:,t)*H)*P;
    end
    %%%%
```

数据的一种方式。摄像头采集的是图像信息，与雷达不同的是，摄像头获取的视频图像数据需要经过数字图像处理算法，检测并提取目标，最终获取目标的颜色、角度、位置等信息，利用这些信息实现对目标的检测和跟踪。在本节中，首先介绍视频图像处理中常用的几个函数，并用它们来对视频做简单的处理。读者可以参考视频图像处理的书籍，以便更深入掌握视频目标检测方法。

3.6.1　视频图像处理的基本方法

本节主要介绍 MATLAB 环境下的视频处理和跟踪算法研究。要在 MATLAB 下研究视频目标跟踪算法，首先要掌握最基本的视频处理方法，包括视频捕获和录制，视频导入和显示，对视频各帧操作，对每一帧中的像素处理的方法。只有掌握这些方法，才能开展视频目标跟踪的工作。

1. 视频捕获和录制

要实现对视频目标跟踪，首先要获取视频。在视频监控领域，很多图像数据都是实时采集的。那么在 MATLAB 环境下如何实现对视频进行实时采集呢？在这里可以利用 MATLAB 视频工具箱，它利用微软公司 Windows 操作系统提供的 VFW（Video For Windows）库函数来对 USB 摄像头进行操作。VFW 用于 Windows 环境下实现实时视频捕获。AVICAP.DLL 模块是 VFW 的一个重要组成部分，它的主要作用是实现视频捕捉。AVICAP 为应用程序提供了一个简单的、基于消息的接口，程序可以访问视频和波形音频硬件，并控制视频流到硬件的捕获。AVICAP 支持实时视频捕获和单帧捕获，并提供对视频流的控制。它能直接访问视频缓冲区，而不需要生成中间文件，实时性很强，效率高。在进行视频捕获之前需要创建一个捕获窗类，它是所有捕获操作及其设置的基础。

很多视频虽然是 AVI 格式的，但由于其编码方式不同导致 MATLAB 无法打开，而用 MATLAB 录制的 AVI 视频则不存在这个问题。通过 MATLAB 捕获和录制 AVI 视频过程中，有一个很重要的函数必须介绍，即 videoinput。它有三个重要的输入参数，分别是 adaptorname、deviceID、format，如图 3.16 所示，其他参数则可以有选择性地设置。deviceID 如果不设置的话，系统会寻找第一个可用的图像采集设备，并使用它。如果计算机上安装了多个摄像头，那么就要设置它们的 ID 号。format 是视频格式，YUV 是视频格式之一，而 YUY2 则是 YUV 中的一种。

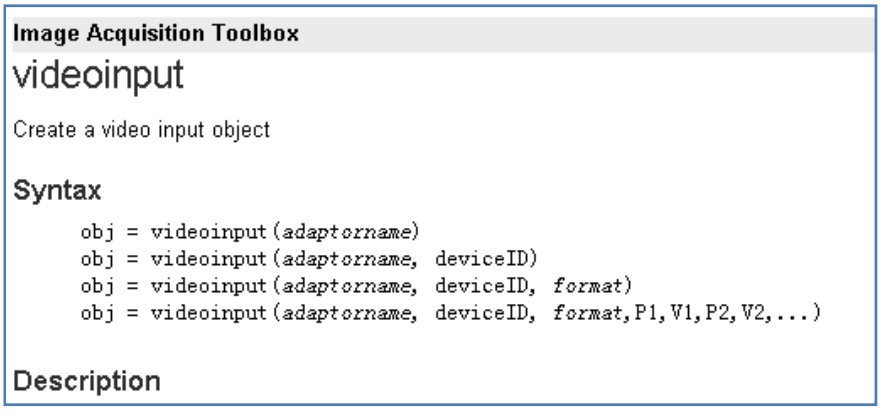


Image Acquisition Toolbox

videoinput

Create a video input object

Syntax

```
obj = videoinput(adaptorname)
obj = videoinput(adaptorname, deviceID)
obj = videoinput(adaptorname, deviceID, format)
obj = videoinput(adaptorname, deviceID, format,P1,V1,P2,V2,...)
```

Description

图 3.16　videoinput 函数

在不知道计算机里安装采集设备的名称情况下，可以在窗口命令行输入 info=imaqhwinfo，查看已安装的图像采集设备，这里可以看到计算机上安装了一个图像采集设备,它的名字是 winvideo，如图 3.17 所示。

```
>>  info=imaqhwinfo

info =

    InstalledAdaptors: {'winvideo'}
        MATLABVersion: '7.1 (R14SP3)'
          ToolboxName: 'Image Acquisition Toolbox'
       ToolboxVersion: '1.9 (R14SP3)'
```

图 3.17　imaqhwinfo

结合以上的介绍，我们通过程序来看看 MATLAB 视频捕获并录制 AVI 视频的过程。录制程序如下所示。

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 程序说明：这是一个视频捕获并录制的程序
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function VideoCapture
%生成一个 AVI 对象，并取名字 myAVI
aviobj = avifile('myAVI');
%创建一个视频输入对象，
```

```
%该对象是你计算机图像采集设备到 matlab 之间的接口
obj=videoinput('winvideo',1,'YUY2_320x240');
%预览视频
preview(obj);
% 设置视频录制的帧数（时间），以便让录制停止
T=100;
k=0;
while (k<T)
    %捕获图像并显示在左边小窗口中
    frame=getsnapshot(obj);
    subplot(1,2,1)
    imshow(frame);

    %转成彩色,显示在右边，这个 frame 就可以按照自己意愿处理了
    frameRGB=ycbcr2rgb(frame);
    subplot(1,2,2);
    imshow(frameRGB);
    drawnow;

    % 将每一帧图像保存在创建的 AVI 视频里面
    aviobj=addframe(aviobj,frameRGB);
    flushdata(obj)

    % 序列自增
    k=k+1
end
% 关闭 avi 对象，同时将截取的图像统一写入 avi 文件中
aviobj = close(aviobj);
%删除对象
delete(obj);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

以上程序运行时可以看到一个视频预览窗口（注意运行上面程序，请确保你已经连接上摄像头，否则将看不到录制的视频）、一个录制显示窗口。程序运行结束，可以在 C:\Program Files\MATLAB71\work 目录下看到录制的 AVI 视频。

2. 视频导入和显示

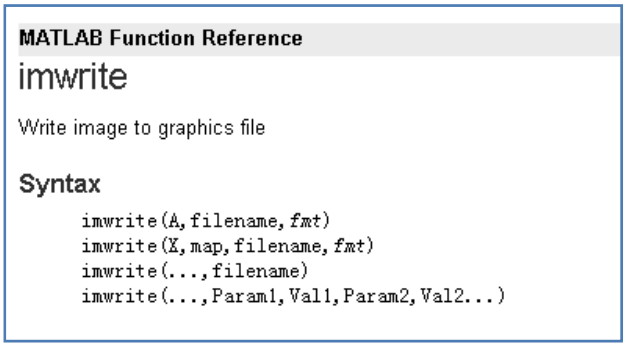
MATLAB 函数库里主要有 aviinfo、avifile、movie，aviread 几个函数用于视频

的导入和显示。下面简要介绍 aviread 函数。

读者需要准备一段 AVI 格式的视频，放置在 C:\Program Files\MATLAB71\work\ 目录下，并且将该段视频的名称命名为 video.avi。如果放在其他位置，请读者修改以下程序中的文件路径。读取并显示 AVI 格式的视频示例程序如下所示。

```
%%%
```

以是 bmp、gif、jpg 等。


MATLAB Function Reference

imwrite

Write image to graphics file

Syntax

```
imwrite(A,filename,fmt)
imwrite(X,map,filename,fmt)
imwrite(...,filename)
imwrite(...,Param1,Val1,Param2,Val2...)
```

图 3.18　imwrite 函数

示例程序如下。

```
%%%
```

运行程序，可以在 C:\Program Files\MATLAB71\work\imageFrame 文件夹下看到一组命名为 image1，image2，…，image100 的图像序列。这个过程就是读取视频帧，并将视频序列帧保存成一幅一幅的图像，命名时统一为“image+序号”的形式。

4. 对帧中的像素操作

该部分是视频图像处理的关键，如果不能对视频中的像素操作，就无法实现目标检测，无法从图像中获取目标位置、角度、颜色等信息。如果读者具备图像处理方面的有关知识，那么理解下面的过程应该会容易。如果读者不具备数字图像处理方面的知识，请参阅专业的图像处理书籍，掌握对灰度图像、彩色图像、二值化、边缘提取、帧间差等基本原理。为了让读者有个感性认识，这里给出一个实例，实现对视频各帧中的画面打上一个“十”字的 logo，读者对其中的像素设为 0 或者 255，看看它是否成为一片白色或者为黑色区域。

读者需要在 C:\Program Files\MATLAB71\work 目录下准备一段名称为 video.avi 的视频和一张名为 1.png 的“+”图片，示例程序如下。

```
%%%
```

```
        subplot(1,2,1);
        imshow(frameData)
        xlabel('The original video')

        % 对该帧开始做手术
        for ii=1:height
            for jj=1:width
                for kk=1:channel
                    %像素复制（替换）
                    frameData(ii,jj,kk)=imageLogo(ii,jj,kk);
                    % 读者可以尝试改成其他值，如下：
                    % frameData(ii,jj,kk)=255;
                end
            end
        end
        %将处理过的视频放在右边
        subplot(1,2,2);
        imshow(frameData)
        xlabel('The processed video')

        % 需要暂停一下，否则画面太快，会感觉是静止的
        pause(0.02);
    end
    %%%%
```

3.6.2 Kalman 滤波对自由下落的皮球跟踪应用

有了 3.6.1 节的视频图像处理基础，现在可以对自由下落的皮球视频图像进行跟踪处理了。对于如图 3.20 所示的自由下落的皮球，要在视频中检测到目标，主要检测目标中心，即红心皮球的重心。在模型建立时可以将该重心抽象成为一个质点，坐标为(x,y)。

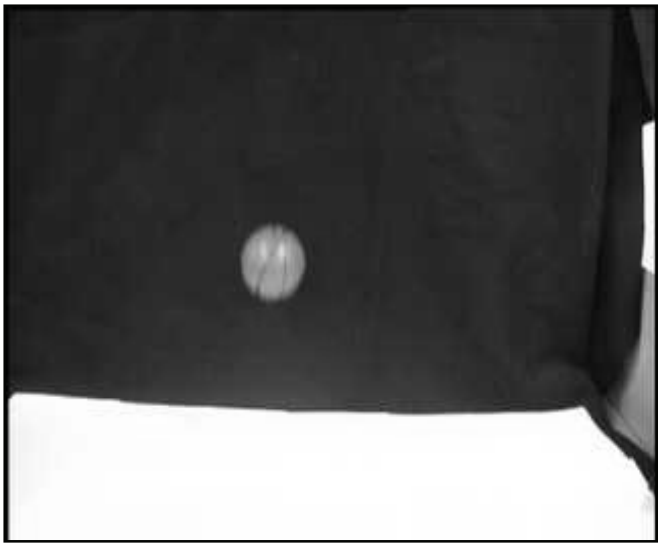


图 3.20　下落的球

定义皮球下落过程中的状态为$\boldsymbol{X}(k)=[x \ \ y \ \ \dot{x} \ \ \dot{y}]^{\mathrm{T}}$，状态方程如下。

$$\boldsymbol{X}(k+1)=\begin{bmatrix}1 & \mathrm{d}t & 0 & 0\\ 0 & 1 & \mathrm{d}t & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\boldsymbol{X}(k)+\begin{bmatrix}0\\0\\0\\g\end{bmatrix}W(k) \tag{3.77}$$

对于状态方程，是典型的位置-速度模型，其中 dt 为采样时间间隔。该模型表明，仅在y方向的速度受到过程噪声的干扰，其他分量的过程噪声为 0。

有了图像信息，在背景不复杂且没有杂物干扰的视频画面，我们可以利用最简单的帧差法获得图像中运动目标，前提是保证摄像机是固定的，仅允许摄像机视野内的目标在运动，这样帧差法非常有效，提取的目标位置信息也非常准确。很显然，我们需要将视频前后 2 帧作差，得到目标的位置信息，那么观测方程可以写为

$$\boldsymbol{Z}(k)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}\boldsymbol{X}(k)+V(k) \tag{3.78}$$

在这个过程中，找到质心(x,y)的过程与雷达目标跟踪中观测目标位置类似，可以对比式（3.64）和式（3.65），它们的状态方程和观测方程几乎是一致的。对于复杂的背景，或者摄像头拍摄画面时，摄像头也是动态的，读者需要建立好相对运动模型。同时对于有干扰的，背景杂物多的视频画面，则需要借助更复杂的目标运动检测算法，在此不再赘述。图 3.21 和图 3.22 是一个利用帧差法检测运动中的皮球，并计算其位置对其跟踪的实例。

图 3.21　检测下落的球

图 3.22　跟踪下落的球

3.6.3 目标检测 MATLAB 程序

```
%%%
```

```
figure
plot(cr,'-g*')
hold on
plot(cc,'-r*')
%%%
```

```
                tmp = id(i);
                id(i) = id(j);
                id(j) = tmp;
            end
        end
    end
    % 确保至少有一个较大的区域（具体如下，最大区域面积要大于 100）
    if stats(1).Area < 100
        return
    end
    selected = (labeled==id(1));
    % 计算最大区域的中心和半径
    centroid = stats(1).Centroid;
    radius = sqrt(stats(1).Area/pi);
    cc = centroid(1);
    cr = centroid(2);
    flag = 1; % 检测到目标，将标志设置为 1
    return
```

3.6.4 Kalman 滤波视频跟踪 MATLAB 程序

```
%%%
```

```
Q=0.01*eye(4);                                  % 过程噪声方差初始化
P = 100*eye(4);                                 % 协方差初始化
dt=1;                                           % 采样时间间隔
A=[[1,0,0,0]',[0,1,0,0]',[dt,0,1,0]',[0,dt,0,1]']; % 状态转移矩阵
g = 6;                  % pixels^2/time step
Bu = [0,0,0,g]';        % 过程噪声驱动矩阵，也即加速度
kfinit=0;               % Kalman 滤波器初始化标志位
x=zeros(100,4);
% 遍历所有图像帧
for i = 1 : 60
    % 读取图像帧
    Im = (imread(['DATA/',int2str(i), '.jpg']));
    imshow(Im)
    imshow(Im)
    Imwork = double(Im);
    % 调用目标检测函数，提取视频中的球
    [cc(i),cr(i),radius,flag] = extractball(Imwork,Imback,i);
    if flag==0          % 没检测到球，跳到下一帧图像
        continue
    end
    hold on
    for c = -1*radius: radius/20 : 1*radius
        r = sqrt(radius^2-c^2);
        plot(cc(i)+c,cr(i)+r,'g.')
        plot(cc(i)+c,cr(i)-r,'g.')
end
    % Kalman 滤波算法
    i                   % 显示图像帧的进度
if kfinit==0            % 如果 Kalman 滤波标志位为 0，说明状态没有初始化
    xp = [MC/2,MR/2,0,0]' ;         % 给出初始状态
else
        xp=A*x(i-1,:)' + Bu;        % 状态预测
end
kfinit=1;               % 状态初始化以后，将标志位置成 1
    PP = A*P*A' + Q ;               % 协方差预测
    K = PP*H'*inv(H*PP*H'+R);       % Kalman 增益
    x(i,:) = (xp + K*([cc(i),cr(i)]' - H*xp))';     % 目标状态更新
    P = (eye(4)-K*H)*PP;            % 协方差更新
    hold on
```

```
        for c = -1*radius: radius/20 : 1*radius
            r = sqrt(radius^2-c^2);
            % 红色的为 Kalman 滤波器估计的位置
            % 以目标区域的中心及半径 radius 画圆，表示跟踪区域
            plot(x(i,1)+c,x(i,2)+r,'r.')    ;
            plot(x(i,1)+c,x(i,2)-r,'r.') ;
        end
        pause(0.3);
    end
    %%%%
```

```
cc = 0;          %  目标中心 x 坐标
cr = 0;          %  目标中心 y 坐标
radius = 0;      %  目标区域半径
flag = 0;        %  检测到目标标志
[MR,MC,Dim] = size(Imback);
%  除去背景，找到最大的不同区域（也即目标区域）
fore = zeros(MR,MC);
%    背景相减，得到目标
fore = (abs(Imwork(:,:,1)-Imback(:,:,1)) > 10) ...
      | (abs(Imwork(:,:,2) - Imback(:,:,2)) > 10) ...
      | (abs(Imwork(:,:,3) - Imback(:,:,3)) > 10);
%  图像腐蚀，除去微小的白噪声点
% bwmorph 该函数的功能是：提取二进制图像的轮廓
foremm = bwmorph(fore,'erode',2);      %  “2”为次数
%  选取最大的目标
labeled = bwlabel(foremm,4);       %  黑背景中甄别有多少白块块，4-联通（上下左右）
% labeled  是标记矩阵，图像分割后对不同的区域进行不同的标记
stats = regionprops(labeled,['basic']);     %basic mohem nist
[N,W] = size(stats);
if N < 1
      return            %  一个目标区域也没检测到就返回
end
%  在 N 个区域中，冒泡算法（从大到小）排序
id = zeros(N);
for i = 1 : N
      id(i) = i;
end
%  将检测到的目标区域按照从大到小排列（冒泡排序）
for i = 1 : N-1
      for j = i+1 : N
            if stats(i).Area < stats(j).Area
                  %  冒泡算法程序
                  tmp = stats(i);
                  stats(i) = stats(j);
                  stats(j) = tmp;
                  tmp = id(i);
                  id(i) = id(j);
                  id(j) = tmp;
            end
```

```
        end
    end
    % 确保至少有一个较大的区域（具体如下，最大区域面积要大于 100）
    if stats(1).Area < 100
        return  % 如果第一个（也是最大的目标区域）都不符合，则没有检测到目标，返回
    end
    selected = (labeled==id(1));
    % 计算最大区域的中心和半径
    centroid = stats(1).Centroid;
    radius = sqrt(stats(1).Area/pi);
    cc = centroid(1); % 相当于 x,中心坐标位置
    cr = centroid(2); % 相当于 y
    flag = 1; % 检测到目标，将标志设置为 1
    return
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

第 4 章　扩展 Kalman 滤波

第 3 章中给出的 Kalman 滤波能够在线性高斯模型的条件下，对目标的状态做出最优的估计，得到较好的跟踪效果。但是，实际系统总是存在不同程度的非线性，典型的非线性函数关系包括平方关系、对数关系、指数关系、三角函数关系等。有些非线性系统可以近似看成线性系统，但为了精确估计系统的状态，大多数系统则不能仅用线性微分方程描述，如飞机的飞行状态、导弹的制导系统、卫星导航系统等，其中的非线性因素不能忽略，必须建立适用于非线性系统的滤波算法。

对于非线性系统滤波问题，常用的处理方法是利用线性化技巧将其转化为一个近似的线性滤波问题，其中应用最广泛的方法是扩展 Kalman 滤波方法（Extended Kalman Filter，EKF）。扩展 Kalman 滤波建立在线性 Kalman 滤波的基础上，其核心思想是，对一般的非线性系统，首先围绕滤波值 $\hat{\boldsymbol{X}}_k$ 将非线性函数 $f(*)$ 和 $h(*)$ 展开成 Taylor 级数并略去二阶及以上项，得到一个近似的线性化模型，然后应用 Kalman 滤波完成对目标的滤波估计等处理。

EKF 的优点是不必预先计算标称轨迹（过程噪声 $\boldsymbol{W}(k)$ 与观测噪声 $\boldsymbol{V}(k)$ 均为 0 时非线性方程的解），但它只能在滤波误差 $\tilde{\boldsymbol{X}}_k = \boldsymbol{X}_k - \hat{\boldsymbol{X}}_k$ 及一步预测误差 $\tilde{\boldsymbol{X}}_{k,k-1} = \boldsymbol{X}_k - \mathrm{X}_{k-1}$ 较小时才能使用。

4.1　扩展 Kalman 滤波原理

4.1.1　局部线性化

离散非线性系统动态方程可以表示为

$$\boldsymbol{X}(k+1) = \boldsymbol{f}[k, \boldsymbol{X}(k)] + \boldsymbol{G}(k)\boldsymbol{W}(k) \tag{4.1}$$

$$\boldsymbol{Z}(k) = \boldsymbol{h}[k, \boldsymbol{X}(k)] + \boldsymbol{V}(k) \tag{4.2}$$

当过程噪声 $\boldsymbol{W}(k)$ 和观测噪声 $\boldsymbol{V}(k)$ 恒为零时，系统模型（4.1）、（4.2）的解为非线性模型的理论解，又称为“标称轨迹”或“标称状态”，而把非线性系统（4.1）、

（4.2）的真实解称为“真轨迹”或“真状态”。

为了便于数学处理，本节假定没有控制量的输入，过程噪声是均值为零的高斯白噪声，且噪声驱动矩阵$\boldsymbol{G}(k)$是已知的，观测噪声$\boldsymbol{V}(k)$是加性均值为零的高斯白噪声，并假定过程噪声$\boldsymbol{W}(k)$和观测噪声$\boldsymbol{V}(k)$序列彼此独立。

首先，扩展 Kalman 滤波（EKF）利用非线性函数的局部线性特性，将非线性模型局部线性化。

由系统状态方程（4.1），将非线性函数$f(*)$围绕滤波值$\hat{\boldsymbol{X}}(k)$做一阶 Taylor 展开，得

$$\boldsymbol{X}(k+1)\approx \boldsymbol{f}[k,\hat{\boldsymbol{X}}(k)]+\frac{\partial \boldsymbol{f}}{\partial \hat{\boldsymbol{X}}(k)}[\boldsymbol{X}(k)-\hat{\boldsymbol{X}}(k)]+\boldsymbol{G}[\hat{\boldsymbol{X}}(k),k]\boldsymbol{W}(k)$$

令

$$\frac{\partial \boldsymbol{f}}{\partial \hat{\boldsymbol{X}}(k)}=\frac{\partial \boldsymbol{f}[\hat{\boldsymbol{X}}(k),k]}{\partial \hat{\boldsymbol{X}}(k)}\bigg|_{\hat{\boldsymbol{X}}(k)=\boldsymbol{X}(k)}=\boldsymbol{\Phi}(k+1\,|\,k)$$

$$\boldsymbol{f}[\hat{\boldsymbol{X}}(k),k]-\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{X}(k)}\bigg|_{\boldsymbol{X}(k)=\hat{\boldsymbol{X}}(k)}\hat{\boldsymbol{X}}(k)=\boldsymbol{\phi}(k)$$

则状态方程为

$$\boldsymbol{X}(k+1)=\boldsymbol{\Phi}(k+1\,|\,k)\boldsymbol{X}(k)+\boldsymbol{G}(k)\boldsymbol{W}(k)+\boldsymbol{\phi}(k) \tag{4.3}$$

初始值为$\boldsymbol{X}(0)=E[\boldsymbol{X}(0)]$。

同 Kalman 滤波基本方程相比，在已经求得前一步滤波值$\hat{\boldsymbol{X}}(k)$的条件下，状态方程（4.3）增加了非随机的外作用项$\boldsymbol{\phi}(k)$。

由系统状态方程（4.2），将非线性函数$h(*)$围绕滤波值$\hat{\boldsymbol{X}}(k)$做一阶 Taylor 展开，得

$$\boldsymbol{Z}(k)=\boldsymbol{h}[\hat{\boldsymbol{X}}(k\,|\,k-1),k]+\frac{\partial \boldsymbol{h}}{\partial \hat{\boldsymbol{X}}(k)}\bigg|_{\hat{\boldsymbol{X}}(k,k-1)}[\boldsymbol{X}(k)-\hat{\boldsymbol{X}}(k\,|\,k-1)]+\boldsymbol{V}(k)$$

令

$$\frac{\partial \boldsymbol{h}}{\partial \hat{\boldsymbol{X}}(k)}\bigg|_{\boldsymbol{X}(k)=\hat{\boldsymbol{X}}(k)}=\boldsymbol{H}(k)$$

$$\mathrm{y}(k)=\boldsymbol{h}[\hat{\boldsymbol{X}}(k\,|\,k-1),k]-\frac{\partial \boldsymbol{h}}{\partial \hat{\boldsymbol{X}}(k)}\bigg|_{\boldsymbol{X}(k)=\hat{\boldsymbol{X}}(k)}\hat{\boldsymbol{X}}(k\,|\,k-1)$$

则观测方程为

$$\boldsymbol{Z}(k)=\boldsymbol{H}(k)\boldsymbol{X}(k)+y(k)+\boldsymbol{V}(k) \tag{4.4}$$

4.1.2　线性 Kalman 滤波

对线性化后的模型（4.3）、（4.4）应用 Kalman 滤波基本方程可得扩展 Kalman 滤波递推方程

$$\hat{\boldsymbol{X}}(k\,|\,k+1)=\boldsymbol{f}(\hat{X}(k\,|\,k)) \tag{4.5}$$

$$\boldsymbol{P}(k+1\,|\,k)=\boldsymbol{\Phi}(k+1\,|\,k)\boldsymbol{P}(k\,|\,k)\boldsymbol{\Phi}^{\mathrm{T}}(k+1\,|\,k)+\boldsymbol{Q}(k+1) \tag{4.6}$$

$$\boldsymbol{K}(k+1)=\boldsymbol{P}(k+1\,|\,k)\boldsymbol{H}^{\mathrm{T}}(k+1)[\boldsymbol{H}(k+1)\boldsymbol{P}(k+1\,|\,k)\boldsymbol{H}^{\mathrm{T}}(k+1)+\boldsymbol{R}(k+1)]^{-1} \tag{4.7}$$

$$\hat{\boldsymbol{X}}(k+1\,|\,k+1)=\hat{\boldsymbol{X}}(k+1\,|\,k)+\boldsymbol{K}(k+1)[\boldsymbol{Z}(k+1)-\boldsymbol{h}(\hat{\boldsymbol{X}}(k+1\,|\,k)] \tag{4.8}$$

$$\boldsymbol{P}(k+1)=[\boldsymbol{I}-\boldsymbol{K}(k+1)\boldsymbol{H}(k+1)]\boldsymbol{P}(k+1\,|\,k) \tag{4.9}$$

式中，滤波初值和滤波误差方差矩阵的初值分别为

$$\boldsymbol{X}(0)=\mathrm{E}[\boldsymbol{X}(0)],\quad \boldsymbol{P}(0)=\mathrm{var}[\boldsymbol{X}(0)]$$

同 Kalman 滤波基本方程相比，在线性化后的系统方程中，状态转移$\boldsymbol{\Phi}(k+1\,|\,k)$和观测矩阵 $\boldsymbol{H}(k+1)$ 由 f 和 h 的雅可比矩阵代替。假设状态变量有 n 维，即 $\boldsymbol{X}=\begin{bmatrix}x_1 & x_2 & \cdots & x_n\end{bmatrix}^{\mathrm{T}}$，则相应雅可比矩阵的求法如下：

$$\boldsymbol{\Phi}(k+1)=\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{X}}=\begin{bmatrix}\dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \cdots & \dfrac{\partial f_1}{\partial x_n}\\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \cdots & \dfrac{\partial f_2}{\partial x_n}\\ \vdots & \vdots & \vdots & \vdots\\ \dfrac{\partial f_n}{\partial x_1} & \dfrac{\partial f_n}{\partial x_2} & \cdots & \dfrac{\partial f_n}{\partial x_n}\end{bmatrix} \tag{4.10}$$

$$\boldsymbol{H}(k+1)=\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{X}}=\begin{bmatrix}\dfrac{\partial h_1}{\partial x_1} & \dfrac{\partial h_1}{\partial x_2} & \cdots & \dfrac{\partial h_1}{\partial x_n}\\ \dfrac{\partial h_2}{\partial x_1} & \dfrac{\partial h_2}{\partial x_2} & \cdots & \dfrac{\partial h_2}{\partial x_n}\\ \vdots & \vdots & \vdots & \vdots\\ \dfrac{\partial h_n}{\partial x_1} & \dfrac{\partial h_n}{\partial x_2} & \cdots & \dfrac{\partial h_n}{\partial x_n}\end{bmatrix} \tag{4.11}$$

4.2 简单非线性系统的扩展 Kalman 滤波器设计

4.2.1 原理介绍

为说明扩展 Kalman 滤波问题，选用下面的标量系统进行分析。系统状态方程为

$$\boldsymbol{X}(k)=0.5\boldsymbol{X}(k-1)+\frac{2.5\boldsymbol{X}(k-1)}{1+\boldsymbol{X}^2(k-1)}+8\cos(1.2k)+W(k) \tag{4.12}$$

观测方程为

$$\boldsymbol{Y}(k)=\frac{\boldsymbol{X}^2(k)}{20}+V(k) \tag{4.13}$$

式（4.12）是包含分式关系、平方关系、三角函数关系的非线性方程，$W(k)$ 为过程噪声，其均值为零、方差为 $\boldsymbol{Q}$。观测方程（4.13）中，观测信号 $\mathrm{Y}(k)$ 与状态 $\mathrm{X}(k)$ 的关系也是非线性的，$V(k)$ 也是均值为零、方差为 $\boldsymbol{R}$ 的高斯白噪声。因此这个系统是一个典型的状态和观测都为非线性的系统。本节以这个典型非线性系统为例，分析如何用扩展 Kalman 滤波来处理噪声。

第一步：初始化初始状态 $\mathrm{X}(0)$、$\mathrm{Y}(0)$、协方差矩阵 $\boldsymbol{P}_0$。

第二步：状态预测

$$X(k\mid k-1)=0.5X(k-1)+\frac{2.5X(k-1)}{1+X^2(k-1)}+8\cos(1.2k) \tag{4.14}$$

第三步：观测预测

$$Y(k\mid k-1)=\frac{X^2(k\mid k-1)}{20} \tag{4.15}$$

第四步：一阶线性化状态方程，求解状态转移矩阵 $\boldsymbol{\Phi}(k)$。

$$\boldsymbol{\Phi}(k)=\frac{\partial f}{\partial X}=0.5+\frac{2.5[1-X^2(k\mid k-1)]}{[1+X^2(k\mid k-1)]^2} \tag{4.16}$$

第五步：一阶线性化观测方程，求解观测矩阵 $H(k)$

$$H(k)=\frac{\partial h}{\partial X}=\frac{X(k\mid k-1)}{10} \tag{4.17}$$

第六步：求协方差矩阵预测 $P(k\mid k-1)$

$$P(k|k-1)=\Phi(k)P(k-1|k-1)\Phi^{\mathrm{T}}(k)+Q \tag{4.18}$$

第七步：求 Kalman 滤波增益

$$K(k)=P(k|k-1)H^{\mathrm{T}}(k)(H(k)P(k|k-1)H^{\mathrm{T}}(k)+R) \tag{4.19}$$

第八步：求状态更新

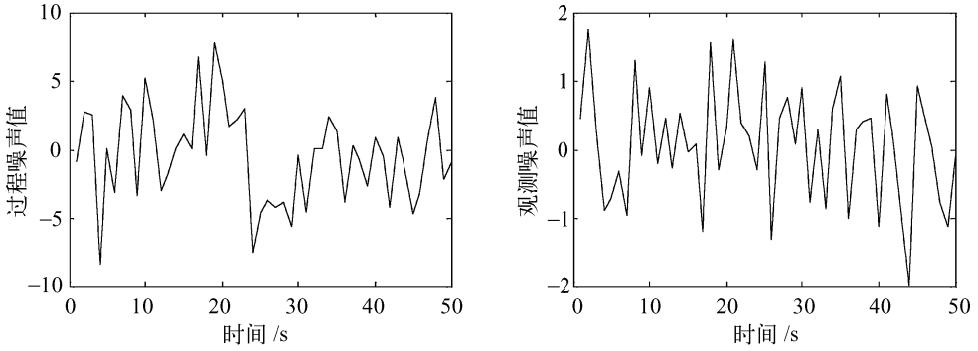
$$X(k)=X(k|k-1)+K(Y(k)-Y(k|k-1)) \tag{4.20}$$

第九步：协方差更新

$$P(k)=(I_n-K(k)H(k))P(k|k-1) \tag{4.21}$$

以上九步为扩展 Kalman 滤波器设计的一个计算周期，各个时刻 EKF 对非线性系统的处理就是这个计算周期不断循环的过程。

在仿真过程中，我们设时间总长度为 50s，过程噪声方差 Q=10，观测噪声方差 R=1，噪声各时刻的大小如图 4.1 所示。本例中的过程噪声偏大，随机扰动强烈，这对 Kalman 滤波来说是个挑战。



（a）过程噪声　　（b）观测噪声

图 4.1　噪声各时刻的大小

初值 $X(0)$=0.1，初值的协方差 $P(0)$=1。对以上系统仿真，得到状态滤波结果如图 4.2 所示。

我们将真实状态与 Kalman 估计的状态作差求绝对值，即 RMS(k)=|$X_{\mathrm{real}}(k)$-$X_{\mathrm{ekf}}(k)$|，得到各个时刻的状态估计偏差如图 4.3 所示。

从估计偏差看，Kalman 滤波几乎没什么效果，这是因为过程噪声太大，随机扰动性增大导致“无规律可循”，最终使滤波器无法降低噪声。但是如果将过程噪声设置为 Q=0.1，这时候可以很明显看到 Kalman 滤波能大大降低噪声，使估计偏差大大减小，如图 4.4 所示。

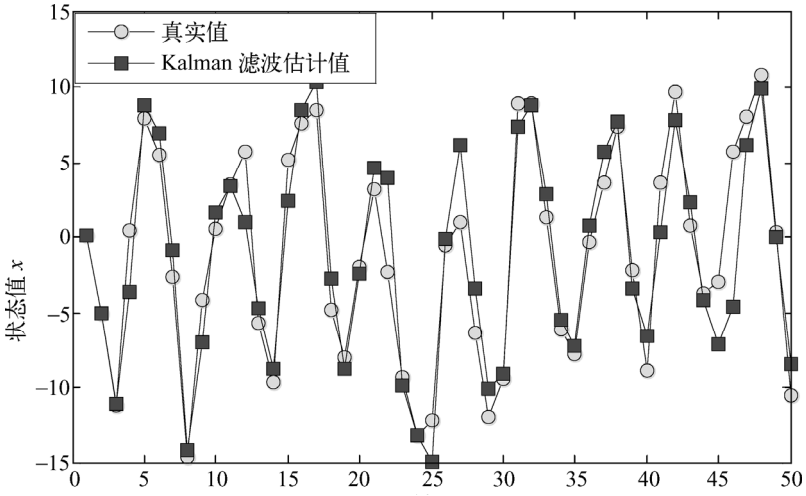


图 4.2　EKF 滤波处理后状态与真值的对比

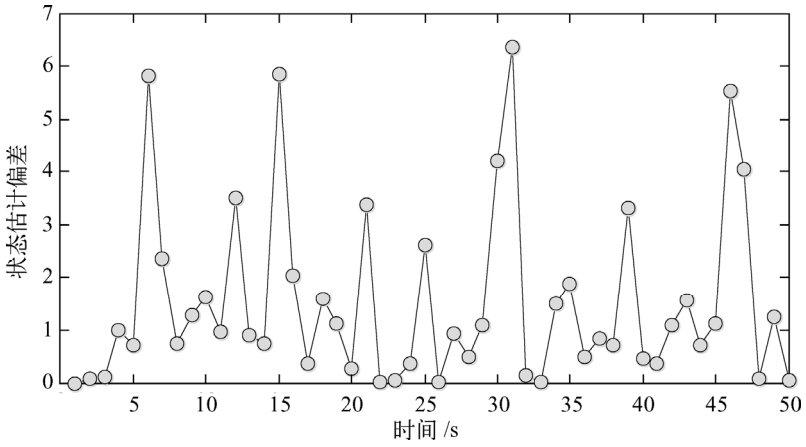


图 4.3　EKF 滤波误差 Q=10

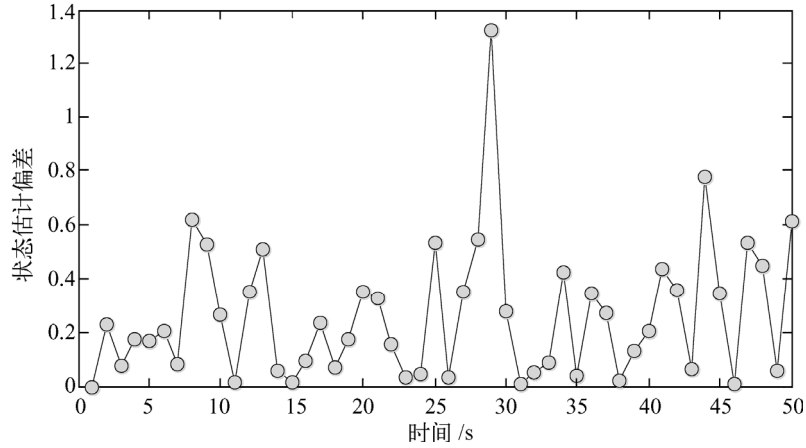


图 4.4　EKF 滤波误差 Q=0.1

4.2.2　标量非线性系统 EKF 的 MATLAB 程序

```
%%%
```

```
        % 求观测矩阵
        H=Xn/10;
        % 协方差预测
        P=F*P0*F'+Q;
        % 求 Kalman 增益
        K=P*H'*inv(H*P*H'+R);
        % 状态更新
        Xekf(k)=Xn+K*(y(k)-Zn);
        % 协方差阵更新
        P0=(eye(1)-K*H)*P;
    end
    % 误差分析
    Xstd=zeros(1,T);
    for k=1:T
        Xstd(k)=abs( Xekf(k)-x(k) );
    end
    %%%%
```

过程，也就是数学建模的过程。

假定目标做匀速直线运动，运动速度为 v，目标在 k 时刻的位置设为 $s(k)$，那么经过采样时间 T，目标的位置则为 $s(k+1)=s(k)+vT$。很显然，在直角坐标系中目标有 x 和 y 方向的分量，运动系统的状态量包括 x 方向的位置、y 方向的位置、x 方向的速度和 y 方向的速度。目标运动过程中受到的随机扰动表示为 $\boldsymbol{U}(k)$，可以将系统表示为

$$\begin{cases} x(k+1)=x(k)+v_x(k)T+\dfrac{1}{2}u_x(k)T^2 \\ v_x(k+1)=v_x(k)+u_x(k)T \\ y(k+1)=y(k)+v_yT+\dfrac{1}{2}u_y(k)T^2 \\ v_y(k+1)=v_y(k)+u_y(k)T \end{cases}$$

为了表示方便，我们常常将系统的状态写为

$$\boldsymbol{X}(k+1)=[x(k),\dot{x}(k),y(k),\dot{y}(k)]^{\mathrm{T}}$$

则系统的状态方程为

$$\boldsymbol{X}(k+1)=\boldsymbol{\Phi}\boldsymbol{X}(k)+\boldsymbol{\Gamma}\mathrm{U}(k) \tag{4.22}$$

式中，

$$\boldsymbol{\Phi}=\begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix},\quad \boldsymbol{\Gamma}=\begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix}$$

观测站或者传感器对目标进行探测，以雷达探测为例，假如雷达站的位置为（x_0,y_0），目标 k 时刻的位置为（$x(k),y(k)$），由雷达发射和反射波测量目标与雷达之间的距离为观测量 $\boldsymbol{Z}$，则观测方程为

$$\boldsymbol{Z}(k)=\sqrt{(x(k)-x_0)^2+(y(k)-y_0)^2}+\boldsymbol{V}(k) \tag{4.23}$$

式中，$\boldsymbol{V}(k)$ 是雷达自身的测量误差，其方差为 $\boldsymbol{R}$。

至此，匀速运动的系统模型已经建立，其状态方程和观测方程分别为式（4.22）和式（4.23）。从模型上看，状态方程是线性的，而观测方程是非线性的。

4.3.2　基于观测距离的 EKF 目标跟踪算法

根据运动观测器获得的量测值（如方位、频率、距离等）对目标进行跟踪是目标运动分析领域中的一个经典问题，在很多应用场合可以获得较为精确的距离

信息。例如，在水下弹道测量系统中，水听器测量脉冲到达时刻获得待测目标的距离，可以采用多传感器对目标进行纯距离跟踪定位；在海上监视中，Ingara 雷达具有 EAR 成像和跟踪两种模式，跟踪模式利用 ISAR 所成高分辨距离像的距离信息对海上目标进行搜索和跟踪。因此，研究仅利用距离信息进行目标跟踪具有十分重要的意义。

以目标跟踪系统式（4.22）和式（4.23）为例，在仿真算例中，假定目标做匀速直线运动，初始状态 $X(0)$已经通过航迹起始算法获得。过程噪声 $\boldsymbol{U}(k)$ 的均方差为

$$\boldsymbol{Q}=\begin{bmatrix} w & 0 \\ 0 & w \end{bmatrix}$$

式中，w 为一个可调节的参数，$w<<1$。测量噪声 $\boldsymbol{V}(k)$ 的均方差 $\boldsymbol{R}=5$。

从模型上看，状态方程（4.22）是线性的，无需线性化求解 $\boldsymbol{\Phi}(k)$，而观测方程（4.23）是非线性的，因此，基于距离信息进行目标跟踪是一个非线性估计问题，可以采用扩展 Kalman 滤波器对目标进行跟踪。

根据 4.1.1 节给出的线性化方法可将该非线性方程线性化，根据式（4.11）得到相应的雅可比矩阵

$$\begin{aligned}\boldsymbol{H}&=\frac{\partial \boldsymbol{Z}(k)}{\partial \boldsymbol{X}(k)}=\left[\frac{\partial \boldsymbol{Z}(k)}{\partial x(k)},\frac{\partial \boldsymbol{Z}(k)}{\partial \dot{x}(k)},\frac{\partial \boldsymbol{Z}(k)}{\partial y(k)},\frac{\partial \boldsymbol{Z}(k)}{\partial \dot{y}(k)}\right]\\&=\left[\frac{x(k)-x_0}{\sqrt{(x(k)-x_0)^2+(y(k)-y_0)^2}},0,\frac{y(k)-y_0}{\sqrt{(x(k)-x_0)^2+(y(k)-y_0)^2}},0\right]\end{aligned} \tag{4.24}$$

综合式（4.22）、式（4.23）和式（4.24），根据 4.1.2 节给出的 EKF 递推方程，可以方便地编写 MATLAB 仿真程序。仿真结果如图 4.5 和图 4.6 所示。

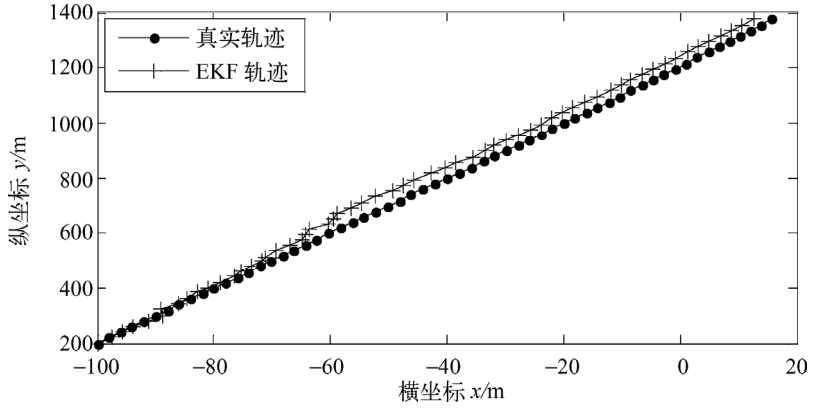


图 4.5　EKF 对运动目标的跟踪轨迹

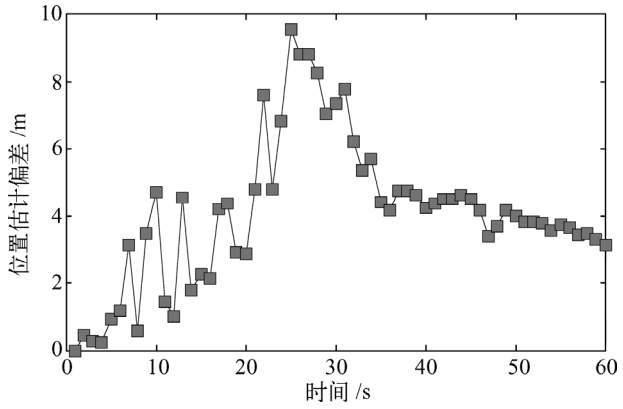


图 4.6　EKF 跟踪误差曲线

4.3.3　基于距离的目标跟踪算法 MATLAB 程序

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 功能说明：扩展 Kalman 滤波在目标跟踪中的应用
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function EKF_For_TargetTracking
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;clear;
T=1;%雷达扫描周期,
N=60/T; %总的采样次数
X=zeros(4,N); % 目标真实位置、速度
X(:,1)=[-100,2,200,20];% 目标初始位置、速度
Z=zeros(1,N); % 传感器对位置的观测
delta_w=1e-3;%如果增大这个参数，目标真实轨迹就是曲线了
Q=delta_w*diag([0.5,1]) ; % 过程噪声方差
G=[T^2/2,0;T,0;0,T^2/2;0,T]; % 过程噪声驱动矩阵
R=5;   %观测噪声方差
F=[1,T,0,0;0,1,0,0;0,0,1,T;0,0,0,1];   % 状态转移矩阵
x0=200;   % 观测站的位置，可以设为其他值
y0=300;
Xstation=[x0,y0];
for t=2:N
    X(:,t)=F*X(:,t-1)+G*sqrtm(Q)*randn(2,1);       %目标真实轨迹
end
for t=1:N
```

```
        Z(t)=Dist(X(:,t),Xstation)+sqrtm(R)*randn;   %对目标观测
    end
    % EKF 滤波
    Xekf=zeros(4,N);
    Xekf(:,1)=X(:,1); % Kalman 滤波状态初始化
    P0=eye(4); %  协方差阵初始化
    for i=2:N
        Xn=F*Xekf(:,i-1); %预测
        P1=F*P0*F'+G*Q*G';%预测误差协方差
        dd=Dist(Xn,Xstation); %  观测预测
        %  求雅可比矩阵 H
        H=[(Xn(1,1)-x0)/dd,0,(Xn(3,1)-y0)/dd,0]; %  即为所求一阶近似
        K=P1*H'*inv(H*P1*H'+R);%增益
        Xekf(:,i)=Xn+K*(Z(:,i)-dd);%状态更新
        P0=(eye(4)-K*H)*P1;%滤波误差协方差更新
    end
    %  误差分析
    for i=1:N
        Err_KalmanFilter(i)=Dist(X(:,i),Xekf(:,i)); %  滤波后的误差
    end
    %%%%
```

```
if length(X2)<=2
    d=sqrt( (X1(1)-X2(1))^2 + (X1(3)-X2(2))^2 );
else
    d=sqrt( (X1(1)-X2(1))^2 + (X1(3)-X2(3))^2 );
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

4.3.4　基于 EKF 的纯方位目标跟踪算法

纯方位目标运动分析是一种隐蔽突击的有效方法，它是利用目标本身的有源辐射，如电磁波辐射、红外辐射、声波辐射、目标对照射的散射，甚至目标施放的干扰辐射等，采用机动单站测向机对目标定位与跟踪。这种对运动目标的隐蔽定位与跟踪是反电子对抗、反水声对抗、反侦察，实施对目标突然袭击的十分有利的手段。在现代战争实际环境里，通常测得的敌机（舰）特征数据是非常有限的，而目标的方位几乎成了唯一可靠的参数，因此可以利用所测得的目标方位角信息估计目标的运动参数（位置、速度、加速度等），从而实施对敌机（舰）进行有效的打击和电子干扰。同 4.3.1 节一样，假定观测站对某匀速直线运动的目标进行纯方位跟踪，假定观测站已知目标的初始状态。

基于纯方位的目标运动模型可以写成如下形式

$$\boldsymbol{X}(k)=\boldsymbol{\Phi}\boldsymbol{X}(k-1)+\boldsymbol{\Gamma}\mathrm{U}(k) \tag{4.25}$$

$$\boldsymbol{Z}(k)=\arctan\left(\frac{y(k)-y_0}{x(k)-x_0}\right)+\boldsymbol{V}(k) \tag{4.26}$$

式中，参数设置与 4.3.1 节相同；$\boldsymbol{V}(k)$ 的均方差 $R=1°$。

从模型上看，状态方程（4.25）是线性的，而观测方程（4.26）是非线性的，根据 4.1.1 节给出的线性化方法可将该非线性方程线性化，根据公式（4.11）得相应的雅可比矩阵为

$$\begin{aligned}\boldsymbol{H}&=\frac{\partial \boldsymbol{Z}(k)}{\partial \mathrm{X}(k)}=\left[\frac{\partial \boldsymbol{Z}(k)}{\partial x(k)},\frac{\partial \boldsymbol{Z}(k)}{\partial \dot{x}(k)},\frac{\partial \boldsymbol{Z}(k)}{\partial y(k)},\frac{\partial \boldsymbol{Z}(k)}{\partial \dot{y}(k)}\right]\\&=\left[\frac{-(y(k)-y_0)}{(x(k)-x_0)^2+(y(k)-y_0)^2},0,\frac{x(k)-x_0}{(x(k)-x_0)^2+(y(k)-y_0)^2},0\right]\end{aligned} \tag{4.27}$$

综合式（4.25）、式（4.26）和式（4.27），根据 4.1.2 节给出的 EKF 递推方程，可以解决纯方位目标跟踪问题，如图 4.7 所示。

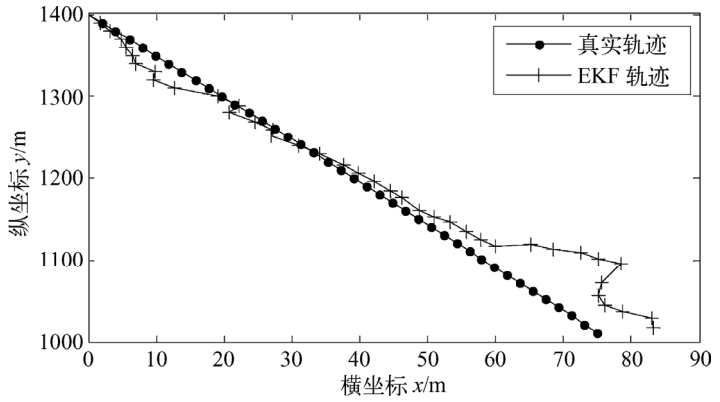


图 4.7　基于 EKF 的纯方位目标跟踪轨迹

这里采用均方根（root mean square，RMS）误差来衡量跟踪的偏差。

$$\mathrm{RMS}=\frac{1}{n}\sqrt{\sum_{i=1}^{n}(x^{*}(i)-x(i))^{2}+(y^{*}(i)-y(i))^{2}} \tag{4.28}$$

EFK 的目标跟踪轨迹以及进一步得到的跟踪偏差 RMS 如图 4.8 所示。

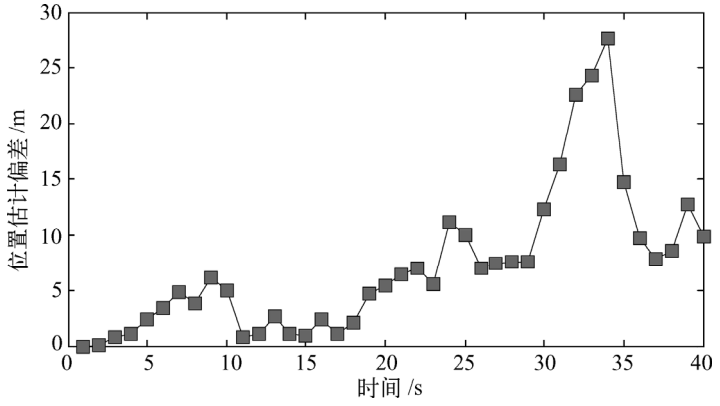
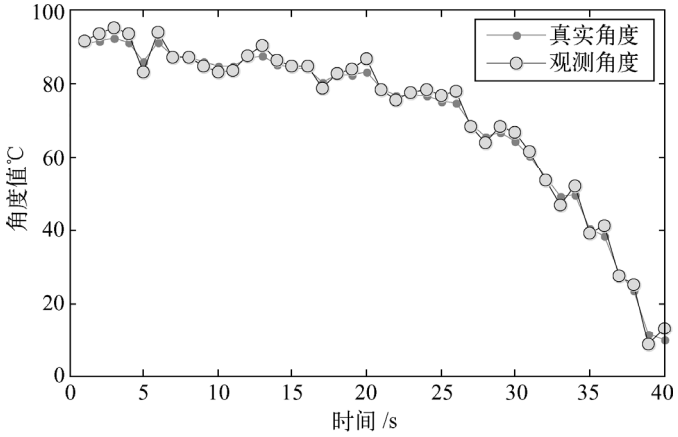


图 4.8　EKF 的跟踪误差 RMS

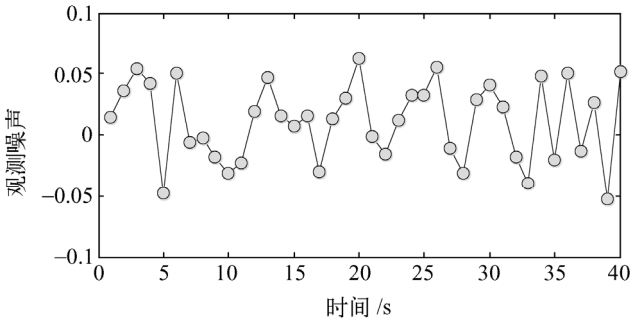
从跟踪轨迹图 4.7 上可以看出 EKF 算法的估计轨迹效果一般，可以说较差，这个主要是因为状态是四维信息，而观测仅是一维角度信息，且角度仅与状态中的 x,y 有非线性关系，这样要让 EKF 算法得到较好的结果是很困难的。

从图 4.8 看，经过几次算法迭代，EKF 估计的误差越来越大，最后 EKF 算法几乎是发散的。在非线性系统中，要根据初始状态和观测信息，达到对目标持续跟踪，是很困难的。对于式（4.25）和（4.26）这样的跟踪模型，系统是非常依赖初始状态的，请读者阅读关于目标跟踪系统的可观测性相关著作就很容易理解这一特点了。

由于受噪声的污染，角度观测值和真实值的对比如图 4.9 所示。本例在仿真中的观测噪声设置得比较大，实际中可能不会出现这种情况。



（a）角度观测值和真实值的对比



（b）观测噪声大小

图 4.9　观测值和真实值对比及观测噪声大小

4.3.5　纯方位目标跟踪算法 MATLAB 程序

```
%%%
```

```
Q=delta_w*diag([1,1]) ;                 % 过程噪声均值
G=[T^2/2,0;T,0;0,T^2/2;0,T];            % 过程噪声驱动矩阵
R=0.1*pi/180;           %观测噪声方差,读者可以修改值观察其对角度测量的影响
F=[1,T,0,0;0,1,0,0;0,0,1,T;0,0,0,1];    % 状态转移矩阵
x0=0;                   % 观测站的位置，可以设为其他值
y0=1000;
Xstation=[x0;y0];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
w=sqrtm(R)*randn(1,N);                          %均值为 0，方差为 1 的高斯噪声
for t=2:N
    X(:,t)=F*X(:,t-1)+G*sqrtm(Q)*randn(2,1);    %目标真实轨迹
end
for t=1:N
    Z(t)=hfun(X(:,t),Xstation)+w(t);            %对目标观测
    %   对 sqrtm(R)*w(t)转化为角度 sqrtm(R)*w(t)/pi*180 可以看出噪声的大小（单位：度）
end
% EKF 滤波
Xekf=zeros(4,N);
Xekf(:,1)=X(:,1);               % Kalman 滤波状态初始化
P0=eye(4);                      % 协方差阵初始化
for i=2:N
    Xn=F*Xekf(:,i-1);           %预测
    P1=F*P0*F'+G*Q*G';          %预测误差协方差
    dd=hfun(Xn,Xstation);       % 观测预测
    % 求雅可比矩阵 H
    D=Dist(Xn,Xstation);
    H=[-(Xn(3,1)-y0)/D,0,(Xn(1,1)-x0)/D,0];     % 即为所求一阶近似
    K=P1*H'*inv(H*P1*H'+R);                     %增益
    Xekf(:,i)=Xn+K*(Z(:,i)-dd);                 %状态更新
    P0=(eye(4)-K*H)*P1;                         %滤波误差协方差更新
end
% 误差分析
for i=1:N
  Err_KalmanFilter(i)=sqrt(Dist(X(:,i),Xekf(:,i)));%滤波后误差
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 画图
```

```
figure
hold on;box on;
plot(X(1,:),X(3,:),'-k.');                    % 真实轨迹
plot(Xekf(1,:),Xekf(3,:),'-r+');              % 扩展 Kalman 滤波轨迹
legend('真实轨迹','EKF 轨迹');
xlabel('横坐标 X/m');
ylabel('纵坐标 Y/m');
figure
hold on; box on;
plot(Err_KalmanFilter,'-ks','MarkerFace','r')
xlabel('时间/s');
ylabel('位置估计偏差/m');
figure
hold on;box on;
plot(Z/pi*180,'-r.','MarkerFace','r');                  % 真实角度值
plot(Z/pi*180+w/pi*180,'-ko','MarkerFace','g');    % 受噪声污染的观测值
legend('真实角度','观测角度');
xlabel('时间/s');
ylabel('角度值/°');
figure          % 观测噪声大小
hold on;box on;
plot(w,'-ko','MarkerFace','g');                         % 受噪声污染的观测值
xlabel('时间/s');
ylabel('观测噪声');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % 子函数
function cita=hfun(X1,X0)                               % 需要注意各个象限角度的变化
if X1(3,1)-X0(2,1)>=0    %    y1-y0>0
      if X1(1,1)-X0(1,1)>0 % x1-x0>0
            cita=atan(abs( (X1(3,1)-X0(2,1))/(X1(1,1)-X0(1,1)) ));
      elseif X1(1,1)-X0(1,1) ==0
            cita=pi/2;
      else
cita=pi/2+atan(abs( (X1(3,1)-X0(2,1))/(X1(1,1)-X0(1,1)) ));
      end
else
      if X1(1,1)-X0(1,1)>0 % x1-x0>0
```

```
cita=3*pi/2+atan(abs( (X1(3,1)-X0(2,1))/(X1(1,1)-X0(1,1)) ));
    elseif X1(1,1)-X0(1,1) ==0
        cita=3*pi/2;
    else
cita=pi+atan(abs( (X1(3,1)-X0(2,1))/(X1(1,1)-X0(1,1)) ));
    end
end
function d=Dist(X1,X2);
if length(X2)<=2
    d=( (X1(1)-X2(1))^2 + (X1(3)-X2(2))^2 );
else
    d=( (X1(1)-X2(1))^2 + (X1(3)-X2(3))^2 );
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

4.4 EKF 在纯方位寻的导弹制导中的应用

4.4.1 三维寻的制导系统

考虑一个在三维平面 *x-y-z* 内运动的质点 *M*，其在某一时刻 *k* 的位置、速度和加速度可用矢量可以表示为

$$\boldsymbol{X}(k)=\left[r_x(k)\quad r_y(k)\quad r_z(k)\quad v_x(k)\quad v_y(k)\quad v_z(k)\quad a_x(k)\quad a_y(k)\quad a_z(k)\right]^{\mathrm{T}}$$

质点 *M* 可以在三维空间内做任何运动，同时假设三个 *x-y-z* 方向上运动具有加性系统噪声 $\boldsymbol{W}(k)$，则在笛卡儿坐标系下该质点的运动状态方程为

$$\boldsymbol{X}(k+1)=f_k(\boldsymbol{X}(k),\boldsymbol{W}(k))$$

通常情况下，上述方程为线性的，即能表示为以下方式。

$$\boldsymbol{X}(k+1)=\boldsymbol{\varphi}\boldsymbol{X}(k)+\boldsymbol{\Gamma}\boldsymbol{U}(k)+\boldsymbol{W}(k) \tag{4.29}$$

式中，

$$\boldsymbol{\varphi}=\begin{bmatrix} I_3 & \Delta t I_3 & \dfrac{1}{\lambda^2}(\mathrm{e}^{-\lambda\Delta t}+\lambda\Delta t-1)I_3 \\ 0_3 & I_3 & \dfrac{1}{\lambda}(1-\mathrm{e}^{-\lambda\Delta t})I_3 \\ 0_3 & 0_3 & \mathrm{e}^{-\lambda\Delta t}I_3 \end{bmatrix},\quad \boldsymbol{\Gamma}=\begin{bmatrix} -(\Delta t^2/2)I_3 \\ -\Delta t I_3 \\ 0_3 \end{bmatrix}$$

Δt 为测量周期，也叫扫描周期、采样时间间隔。动态噪声 $\boldsymbol{W}(k)$ 为

$$\boldsymbol{W}(k)=\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & \omega_x(k) & \omega_y(k) & \omega_z(k)\end{bmatrix}^{\mathrm{T}}$$

且

$$\mathrm{E}\left[\boldsymbol{W}(k)\right]=q_1=0_{9\times1},\quad \mathrm{E}\left[\boldsymbol{W}(k)\boldsymbol{W}^{\mathrm{T}}(k)\right]=\boldsymbol{Q}_1=\begin{bmatrix}0_6 & 0_{6\times3}\\ 0_{3\times6} & \sigma^2 I_3\end{bmatrix}$$

$\boldsymbol{W}(k)$ 是高斯型白色随机向量序列。

考虑一个带有观测器的飞行中的导弹，假设其为质点 M，对移动的目标进行观测，导弹与目标的相对位置依然可用 x-y-z 表示，如图 4.10 所示。

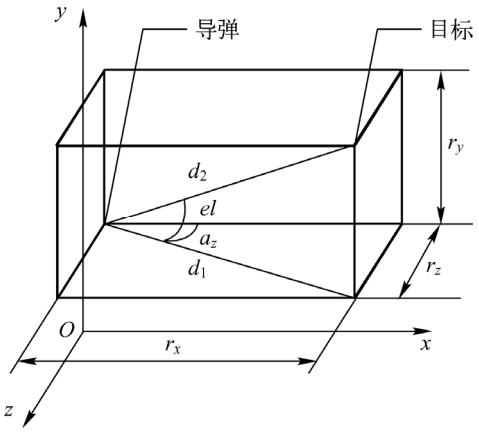


图 4.10　导弹与目标的相对位置示意图

导弹对目标采用纯方位角观测，观测量为俯仰角和水平方向偏向角，实际测量中雷达具有加性测量噪声 $\boldsymbol{V}(k)$，在笛卡儿坐标系下，观测方程为

$$\boldsymbol{Z}(k)=h\left[\boldsymbol{X}(k)\right]+\boldsymbol{V}(k) \tag{4.30}$$

式中，

$$h[\boldsymbol{X}(k)]=\left[\arctan\frac{r_y(k)}{\sqrt{r_x^2(k)+r_z^2(k)}},\ \arctan\frac{-r_x(k)}{r_z(k)}\right]^{\mathrm{T}} \tag{4.31}$$

$\boldsymbol{V}(k)$ 为测量噪声，是高斯型白色随机向量序列，且

$$\mathrm{E}\left[\boldsymbol{V}(k)\right]=r_1=0_{2\times1},\quad \mathrm{E}\left[\boldsymbol{V}(k)\boldsymbol{V}^{\mathrm{T}}(k)\right]=\boldsymbol{R}_1 \tag{4.32}$$

对于 $\boldsymbol{R}_1$，其定义为

$$\boldsymbol{R}_1(k)=\boldsymbol{D}^{-1}(k)\boldsymbol{x}\boldsymbol{D}^{-\mathrm{T}}(k) \tag{4.33}$$

式中，$\boldsymbol{x}=0.1\boldsymbol{I}_2$

$$\boldsymbol{D}(k)=\begin{bmatrix}\sqrt{r_x^2(k)+r_y^2(k)+r_z^2(k)} & 0\\ 0 & \sqrt{r_x^2(k)+r_y^2(k)+r_z^2(k)}\end{bmatrix} \tag{4.34}$$

综合式（4.30）～式（4.34）可知，在笛卡儿坐标系下，该运动模型观测方程是非线性的。

4.4.2 EKF 在寻的制导问题中的算法分析

下面参考国防工业出版社周荻著《寻的导弹新型导引规律》第 5.5 和 5.6 节，结合导弹跟踪的特点，细述 EKF 算法的步骤。

第一步：初始化

设定采样时间，仿真时长

$$\Delta t=0.01\mathrm{s},\qquad t=3.7\mathrm{s}$$

设定导弹的初始状态

$$\boldsymbol{x}(0)=[3500,1500,1000,-1100,-150,-50,10,10,10]^{\mathrm{T}}$$

设定 EKF 滤波器估计的初始化状态

$$\boldsymbol{ex}(0)=[3000,1200,800,-950,-100,-100,0,0,0]^{\mathrm{T}}$$

设定过程噪声方差

$$\sigma^2=0.1,\quad \boldsymbol{Q}=[0_{6\times6},0_{3\times6};0_{6\times3},\sigma^2\times\boldsymbol{I}_{3\times3}]$$

初始化 EKF 滤波器估计的状态协方差矩阵

$$\boldsymbol{P}_0=[10^4\times\boldsymbol{I}_6,\boldsymbol{0}_{6\times3};\boldsymbol{0}_{3\times6},10^2\times\boldsymbol{I}_3]$$

第二步：EKF 滤波估计

for k=2:T----------

Step1：目标运动

x(:,k)=F*x(:,k-1)+G*u(:,k-1)+w(:,k-1);

Step2：每隔 $\Delta t=0.01\mathrm{s}$ 对目标扫描，即观测。

z(:,k)=[atan(x(2,k-1)/sqrt(x(1,k-1)^2+x(3,k-1)^2))，atan(-1*x(3,k-1)/x(1,k-1))]'+v(:,k);

Step3：计算统计方差估计 R，R=0.1*eye(2)/(DD*DD')

Step4：状态预测（结合 Kalman 滤波核心公式），Xn=F*ex+G*u;

Step5：观测预测（结合 Kalman 滤波核心公式），……（参见程序）

Step6：协方差阵预测，P=F*P0*F'+Q;

Step7：这一步是对于 EKF 量身定做的，也是 EKF 的核心，计算线性 $\boldsymbol{H}$ 矩阵：

对于本节讨论的非线性系统（4.29）、（4.30），定义

$$\boldsymbol{f}_k^x = \left.\frac{\partial \boldsymbol{f}_k(X_k)}{\partial \boldsymbol{X}}\right|_{X_k=\hat{X}_{k|k-1}}$$

$$\boldsymbol{h}_k^x = \left.\frac{\partial \boldsymbol{h}_k(X_k)}{\partial \boldsymbol{X}}\right|_{X_k=\hat{X}_{k|k-1}}$$

因为系统状态方程（4.29）为线性的，所以 $\boldsymbol{f}_k^x = \boldsymbol{F}_k$，而观测方程（4.30）为非线性的，对其关于 $\boldsymbol{x}_k$ 求偏导，得

$$\boldsymbol{H} = \frac{\partial \boldsymbol{h}(x(k))}{\partial x} = \begin{bmatrix} \dfrac{-r_x(k)r_y(k)}{\sqrt{r_x^2(k)+r_y^2(k)+r_z^2(k)}} & \dfrac{\sqrt{r_x^2(k)+r_z^2(k)}}{\sqrt{r_x^2(k)+r_y^2(k)+r_z^2(k)}} \\ \dfrac{r_x(k)}{r_x^2(k)+r_z^2(k)} & 0 \\ \dfrac{-r_y(k)r_z(k)}{r_x^2(k)+r_y^2(k)+r_z^2(k)} & 0\ 0\ 0\ 0\ 0\ 0 \\ \dfrac{-r_x(k)}{r_x^2(k)+r_z^2(k)} & 0\ 0\ 0\ 0\ 0\ 0 \end{bmatrix}$$

Step8：计算 Kalman 滤波增益 K=P*H'/(H*P*H'+R);

Step9：状态更新 ex=Xn+K*(z-Zn);

Step10：协方差阵更新，P0=(I-K*H)*P

End　for ----------

4.4.3　仿真结果

运行程序，得到目标的跟踪轨迹如图 4.11 所示。跟踪轨迹最终表明，滤波估计状态较好地跟踪了目标，轨迹趋于一致。

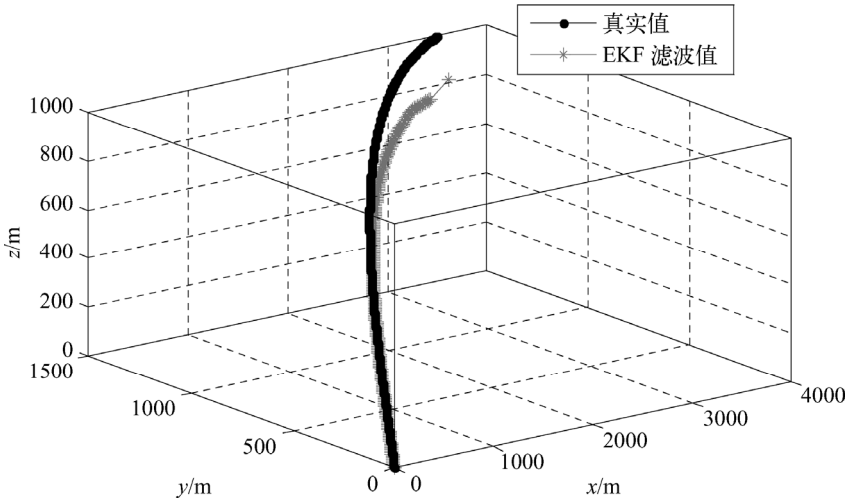


图 4.11　纯方位寻的导弹 EKF 跟踪轨迹与真实轨迹

同样计算 EKF 滤波后的状态值与目标真实状态之间的偏差，可以得到位置偏差图、速度偏差图和加速度偏差图，如图 4.12～图 4.14 所示。无论位置还是速度，最终都是收敛的，加速度则最终稳定在特定的值内。

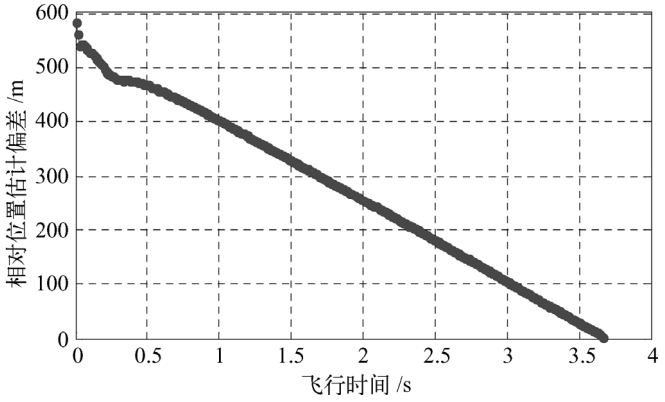


图 4.12　寻的导弹 EKF 跟踪位置偏差

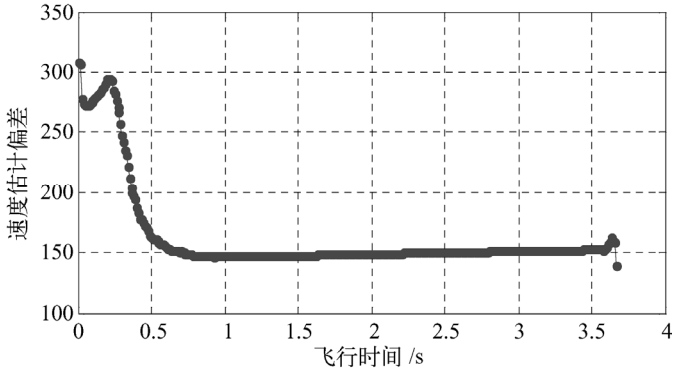


图 4.13　寻的导弹 EKF 跟踪速度偏差

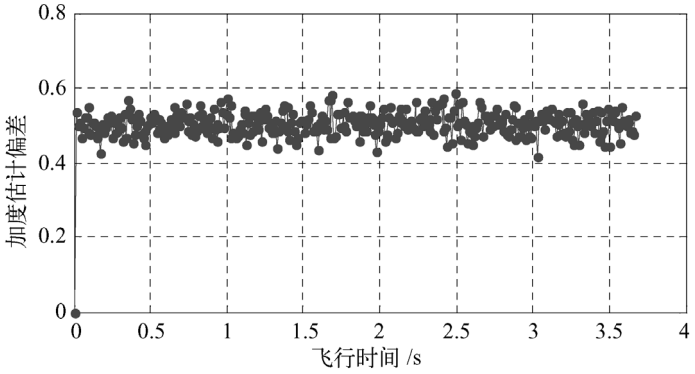


图 4.14　寻的导弹 EKF 跟踪加速度偏差

4.4.4　寻的制导 MATLAB 程序

```
%%%
```

```
ex(:,1)=[3000,1200,960,-800,-100,-100,0,0,0]';% 滤波器状态 Xekf（0）
cigema=sqrt(0.1);
w=[zeros(6,T);cigema*randn(3,T)]; % 过程噪声
Q=[zeros(6),zeros(6,3);zeros(3,6),cigema^2*eye(3)];
z=zeros(2,T);            % 观测值
z(:,1)=[atan( x(2,1)/sqrt(x(1,1)^2+x(3,1)^2) ), atan(-1*x(3,1)/x(1,1))]';
v=zeros(2,T);          % 观测噪声
for k=2:T-3
    tgo=tf-k*0.01+0.0000000000000001;
    c1=N/tgo^2;     % 制导律的系数
    c2=N/tgo;       % 制导律的系数
    c3=N*(exp(-longa*tgo)+longa*tgo-1)/(longa*tgo)^2;   % 制导律的系数
    % X、Y、Z 三个方向的导弹加速度
    u(1,k-1)=[c1,c2,c3]*[x(1,k-1),x(4,k-1),x(7,k-1)]';
    u(2,k-1)=[c1,c2,c3]*[x(2,k-1),x(5,k-1),x(8,k-1)]';
    u(3,k-1)=[c1,c2,c3]*[x(3,k-1),x(6,k-1),x(9,k-1)]';
    x(:,k)=F*x(:,k-1)+G*u(:,k-1)+w(:,k-1);   % 目标状态方程
    d=sqrt(x(1,k)^2+x(2,k)^2+x(3,k)^2);
    D=[d,0;0,d];   % 参考书中公式
    R=inv(D)*0.1*eye(2)*inv(D)';% 观测噪声方差
    v(:,k)=sqrtm(R)*randn(2,1); % 观测噪声模拟
    % 目标观测方程
    z(:,k)=[atan( x(2,k)/sqrt(x(1,k)^2+x(3,k)^2) ), ...
        atan(-1*x(3,k)/x(1,k))]'+v(:,k);
end
% 下面根据观测值开始滤波
P0=[10^4*eye(6),zeros(6,3);zeros(3,6),10^2*eye(3)]; % 协方差初始化
eP0=P0;
stop=0.5/0.01;
span=1/0.01;
for k=2:T-3
    dd=sqrt(ex(1,k-1)^2+ex(2,k-1)^2+ex(3,k-1)^2);
    DD=[dd,0;0,dd];
    RR=0.1*eye(2)/(DD*DD');
    tgo=tf-k*0.01+0.0000000000000001;
    c1=N/tgo^2;
    c2=N/tgo;
    c3=N*(exp(-longa*tgo)+longa*tgo-1)/(longa*tgo)^2;
    u(1,k-1)=[c1,c2,c3]*[ex(1,k-1),ex(4,k-1),ex(7,k-1)]';
```

```
            u(2,k-1)=[c1,c2,c3]*[ex(2,k-1),ex(5,k-1),ex(8,k-1)]';
            u(3,k-1)=[c1,c2,c3]*[ex(3,k-1),ex(6,k-1),ex(9,k-1)]';
            % 调用扩展 Kalman 算法子函数
            [ex(:,k),eP0]=ekf(F,G,Q,RR,eP0,u(:,k-1),z(:,k),ex(:,k-1));
        end
        for t=1:T-3 % 求每个时间点误差的平方
            Ep_ekfx(i,t)=sqrt((ex(1,t)-x(1,t))^2);
            Ep_ekfy(i,t)=sqrt((ex(2,t)-x(2,t))^2);
            Ep_ekfz(i,t)=sqrt((ex(3,t)-x(3,t))^2);
            Ep_ekf(i,t)=sqrt( (ex(1,t)-x(1,t))^2+(ex(2,t)-x(2,t))^2+(ex(3,t)-x(3,t))^2 );
            Ev_ekf(i,t)=sqrt( (ex(4,t)-x(4,t))^2+(ex(5,t)-x(5,t))^2+(ex(6,t)-x(6,t))^2 );
            Ea_ekf(i,t)=sqrt( (ex(7,t)-x(7,t))^2+(ex(8,t)-x(8,t))^2+(ex(9,t)-x(9,t))^2 );
        end

        for t=1:T-3 % 求误差的均值，即 RMS
            error_x(t)=mean(Ep_ekfx(:,t));
            error_y(t)=mean(Ep_ekfy(:,t));
            error_z(t)=mean(Ep_ekfz(:,t));
            error_r(t)=mean(Ep_ekf(:,t));
            error_v(t)=mean(Ev_ekf(:,t));
            error_a(t)=mean(Ea_ekf(:,t));
        end
    end

    t=0.01:0.01:3.67;
    figure  % 轨迹图
    hold on;box on;grid on;
    plot3(x(1,:),x(2,:),x(3,:),'-k.')
    plot3(ex(1,:),ex(2,:),ex(3,:),'-r*','MarkerFace','r')
    legend('真实值','EKF 滤波值');
    view(3)
    xlabel('x/m');
    ylabel('y/m');
    zlabel('z/m');
    figure  % 位置偏差图
    hold on;box on;grid on;
    plot(t,error_r,'-b.');
    xlabel('飞行时间/s');
    ylabel('相对位置估计偏差/m');
```

```
figure      % 速度偏差图
hold on;box on;grid on;
plot(t,error_v,'-b.');
xlabel('飞行时间/s');
ylabel('速度估计偏差');
figure      % 加速度偏差图
hold on;box on;grid on;
plot(t,error_a,'-b.');
xlabel('飞行时间/s');
ylabel('加度估计偏差');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 函数说明：  扩展 Kalman 滤波算法，子函数
% 函数参数：  ex 为扩展 Kalman 估计得到的状态
function [ex,P0]=ekf(F,G,Q,R,P0,u,z,ex)
% 状态预测
Xn=F*ex+G*u;
% 观测预测
Zn=[atan( Xn(2)/sqrt(Xn(1)^2+Xn(3)^2) ),atan(-1*Xn(3)/Xn(1))]';
% 协方差阵预测
P=F*P0*F'+Q;
% 计算线性化的 H 矩阵
dh1_dx=-1*Xn(1)*Xn(2)/(Xn(1)^2+Xn(2)^2+Xn(3)^2)/sqrt(Xn(1)^2+Xn(3)^2);
dh1_dy=sqrt(Xn(1)^2+Xn(3)^2)/(Xn(1)^2+Xn(2)^2+Xn(3)^2);
dh1_dz=-1*Xn(2)*Xn(3)/(Xn(1)^2+Xn(2)^2+Xn(3)^2)/sqrt(Xn(1)^2+Xn(3)^2);
dh2_dx=Xn(3)/(Xn(1)^2+Xn(3)^2);
dh2_dy=0;
dh2_dz=-1*Xn(1)/(Xn(1)^2+Xn(3)^2);
H=[dh1_dx,dh1_dy,dh1_dz,0,0,0,0,0,0;dh2_dx,dh2_dy,dh2_dz,0,0,0,0,0,0];
% Kalman 增益
K=P*H'/(H*P*H'+R);
% 状态更新
ex=Xn+K*(z-Zn);
% 协方差阵更新
P0=(eye(9)-K*H)*P;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

第 5 章　无迹 Kalman 滤波

第 4 章讨论的扩展 Kalman 滤波算法是对非线性的系统方程或者观测方程进行泰勒展开并保留其一阶近似项，这样不可避免地引入了线性化误差。如果线性化假设不成立，采用这种算法则会导致滤波器性能下降以至于造成发散。另外，在一般情况下计算系统状态方程和观测方程的 Jacobian 矩阵是不易实现的，增加了算法的计算复杂度。

无迹 Kalman 滤波（Unscented Kalman Filter，UKF）摒弃了对非线性函数进行线性化的传统做法，采用 Kalman 线性滤波框架，对于一步预测方程，使用无迹变换（Unscented Transform，UT）来处理均值和协方差的非线性传递问题。UKF 算法是对非线性函数的概率密度分布进行近似，用一系列确定样本来逼近状态的后验概率密度，而不是对非线性函数进行近似，不需要对 Jacobian 矩阵进行求导。UKF 没有把高阶项忽略，因此对于非线性分布的统计量有较高的计算精度，有效地克服了扩展 Kalman 滤波的估计精度低、稳定性差的缺陷。

5.1　无迹 Kalman 滤波原理

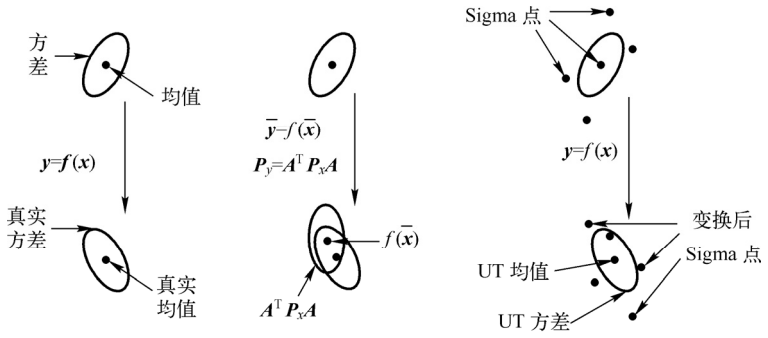
5.1.1　无迹变换

无迹 Kalman 滤波是 S.Julier 等人提出的一种非线性滤波方法。与扩展 Kalman 滤波不同的是，它并不对非线性方程 f 和 h 在估计点处做线性化逼近，而是利用无迹变换在估计点附近确定采样点，用这些样本点表示的高斯密度近似状态的概率密度函数。

UT 变换实现方法为：在原状态分布中按某一规则选取一些采样点，使这些采样点的均值和协方差等于原状态分布的均值和协方差；将这些点代入非线性函数中，相应得到非线性函数值点集，通过这些点集求取变换后的均值和协方差。这样得到的非线性变换后的均值和协方差精度最少具有 2 阶精度（Taylor 序列展开）。

对于高斯分布，可达到 3 阶精度。其采样点的选择是基于先验均值和先验协方差矩阵的平方根的相关列实现的。

非线性变换比较如图 5.1 所示。



（a）真实分布　（b）扩展卡尔曼滤波　（c）UT 变换滤波

图 5.1　非线性变换比较

下面以对称分布采样的 UT 变换为例，简要介绍 UT 变换的基本原理。设一个非线性变换 $y=\boldsymbol{f}(\boldsymbol{x})$。状态向量 $\boldsymbol{x}$ 为 n 维随机变量，并且已知其均值 $\bar{\boldsymbol{x}}$ 和方差 $\boldsymbol{P}$。则可通过下面的 UT 变换得到 $2n+1$ 个 Sigma 点 $\boldsymbol{X}$ 和相应的权值 ω 来计算 $\boldsymbol{y}$ 的统计特征：

（1）计算 $2n+1$ 个 Sigma 点，即采样点，这里的 n 指的是状态的维数。

$$\begin{cases}\boldsymbol{X}^{(0)}=\bar{\boldsymbol{X}},\ i=0\\ \boldsymbol{X}^{(i)}=\bar{\boldsymbol{X}}+(\sqrt{(n+\lambda)\boldsymbol{P}})_i,\ i=1\sim n\\ \boldsymbol{X}^{(i)}=\bar{\boldsymbol{X}}-(\sqrt{(n+\lambda)\boldsymbol{P}})_i,\ i=n+1\sim 2n\end{cases}\tag{5.1}$$

式中，$(\sqrt{\boldsymbol{P}})^{\mathrm{T}}(\sqrt{\boldsymbol{P}})=\boldsymbol{P}$，$(\sqrt{\boldsymbol{P}})_i$ 表示矩阵方根的第 i 列。

（2）计算这些采样点相应的权值

$$\begin{cases}\omega_{\mathrm{m}}^{(0)}=\dfrac{\lambda}{n+\lambda}\\ \omega_{\mathrm{c}}^{(0)}=\dfrac{\lambda}{n+\lambda}+(1-a^2+\beta)\\ \omega_{\mathrm{m}}^{(i)}=\omega_{\mathrm{c}}^{(i)}=\dfrac{\lambda}{2(n+\lambda)},i=1\sim 2n\end{cases}\tag{5.2}$$

式中，下标 m 为均值，c 为协方差，上标为第几个采样点。参数 $\lambda=a^2(n+\kappa)-n$ 是一个缩放比例参数，用来降低总的预测误差，a 的选取控制了采样点的分布状态，

κ为待选参数，其具体取值虽然没有界限，但通常应确保矩阵$(n+\lambda)P$为半正定矩阵。待选参数$\beta \geqslant 0$是一个非负的权系数，它可以合并方程中高阶项的动差，这样就可以把高阶项的影响包括在内。

UT 变换得到的 Sigma 点集具有下述的性质：

（1）由于 Sigma 点集围绕均值对称分布并且对称点具有相同的权值，因此 Sigma 集合的样本均值为$\bar{\boldsymbol{X}}$，与随机向量$\boldsymbol{X}$的均值相同。

（2）对于 Sigma 点集的样本方差与随机向量$\boldsymbol{X}$的方差相同。

（3）任意正态分布的 Sigma 点集，是由标准正态分布的 Sigma 集合经过一个变换得到的。

5.1.2　无迹 Kalman 滤波算法实现

对于不同时刻 k，由具有高斯白噪声$\boldsymbol{W}(k)$的随机变量 $\boldsymbol{X}$ 和具有高斯白噪声$\boldsymbol{V}(k)$的观测变量$\boldsymbol{Z}$构成的非线性系统可以由式（5.3）描述，

$$\begin{cases} \boldsymbol{X}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k),\ \boldsymbol{W}(k)) \\ \boldsymbol{Z}(k) = \boldsymbol{h}(\boldsymbol{x}(k),\ \boldsymbol{V}(k)) \end{cases} \tag{5.3}$$

式中，$\boldsymbol{f}$是非线性状态方程函数；$\boldsymbol{h}$ 是非线性观测方程函数。设$\boldsymbol{W}(k)$具有协方差阵$\boldsymbol{Q}$，$\boldsymbol{V}(k)$具有协方差阵 $\boldsymbol{R}$。随机变量$\boldsymbol{X}$在不同时刻 k 的无迹 Kalman 滤波算法基本步骤如下：

（1）利用式（5.1）和（5.2）获得一组采样点（称为 Sigma 点集）及其对应权值。

$$\boldsymbol{X}^{(i)}(k \mid k) = [\hat{\boldsymbol{X}}(k \mid k) \quad \hat{\boldsymbol{X}}(k \mid k) + \sqrt{(n+\lambda)\boldsymbol{P}(k \mid k)} \quad \hat{\boldsymbol{X}}(k \mid k) - \sqrt{(n+\lambda)\boldsymbol{P}(k \mid k)}]$$

（2）计算 $2n$+1 个 Sigma 点集的一步预测，i=1, 2, …, $2n$+1。

$$\boldsymbol{X}^{(i)}(k+1 \mid k) = \boldsymbol{f}[k, \boldsymbol{X}^{(i)}(k \mid k)]$$

（3）计算系统状态量的一步预测及协方差矩阵，它由 Sigma 点集的预测值加权求和得到，其中权值$\omega^{(i)}$通过式（5.2）得到。这一点不同于传统的 Kalman 滤波算法，传统 Kalman 算法只需通过上一时刻的状态代入状态方程，仅计算一次便获得状态的预测；而 UKF 在此利用一组 Sigma 点的预测，并计算对它们加权求均值，得到系统状态量的一步预测。

$$\hat{\boldsymbol{X}}(k+1|k)=\sum_{i=0}^{2n}\omega^{(i)}\boldsymbol{X}^{(i)}(k+1|k)$$

$$\boldsymbol{P}(k+1|k)=\sum_{i=0}^{2n}\omega^{(i)}[\hat{\boldsymbol{X}}(k+1|k)-\boldsymbol{X}^{(i)}(k+1|k)][\hat{\boldsymbol{X}}(k+1|k)-\boldsymbol{X}^{(i)}(k+1|k)]^{\mathrm{T}}+\boldsymbol{Q}$$

（4）根据一步预测值，再次使用 UT 变换，产生新的 Sigma 点集。

$$\boldsymbol{X}^{(i)}(k+1|k)=[\hat{\mathrm{X}}(k+1|k)\quad\hat{\boldsymbol{X}}(k+1|k)+\sqrt{(n+\lambda)\boldsymbol{P}(k+1|k)}$$
$$\hat{\boldsymbol{X}}(k+1|k)-\sqrt{(n+\lambda)\boldsymbol{P}(k+1|k)}]$$

（5）将由步骤（4）预测的 Sigma 点集代入观测方程，得到预测的观测量，i=1, 2, …, 2n+1。

$$\boldsymbol{Z}^{(i)}(k+1|k)=\boldsymbol{h}[\boldsymbol{X}^{(i)}(k+1|k)]$$

（6）由步骤（5）得到 Sigma 点集的观测预测值，通过加权求和得到系统预测的均值及协方差。

$$\bar{\boldsymbol{Z}}(k+1|k)=\sum_{i=0}^{2n}\omega^{(i)}\boldsymbol{Z}^{(i)}(k+1|k)$$

$$\boldsymbol{P}_{z_kz_k}=\sum_{i=0}^{2n}\omega^{(i)}[\boldsymbol{Z}^{(i)}(k+1|k)-\bar{\boldsymbol{Z}}*k+1|k)][\boldsymbol{Z}^{(i)}(k+1|k)-\bar{\boldsymbol{Z}}*k+1|k)]^{\mathrm{T}}+R$$

$$\boldsymbol{P}_{x_kz_k}=\sum_{i=0}^{2n}\omega^{(i)}[\boldsymbol{X}^{(i)}(k+1|k)-\bar{\boldsymbol{Z}}(k+1|k)][\boldsymbol{Z}^{(i)}(k+1|k)-\bar{\boldsymbol{Z}}(k+1|k)]^{\mathrm{T}}$$

（7）计算 Kalman 增益矩阵。

$$\boldsymbol{K}(k+1)=\boldsymbol{P}_{x_kz_k}\boldsymbol{P}_{z_kz_k}^{-1}$$

（8）最后，计算系统的状态更新和协方差更新。

$$\hat{\boldsymbol{X}}(k+1|k+1)=\hat{\boldsymbol{X}}(k+1|k)+\boldsymbol{K}(k+1)[\boldsymbol{Z}(k+1)-\hat{\boldsymbol{Z}}(k+1|k)]$$

$$\boldsymbol{P}(k+1|k+1)=\boldsymbol{P}(k+1|k)-\boldsymbol{K}(k+1)\boldsymbol{P}_{z_kz_k}\boldsymbol{K}^{\mathrm{T}}(k+1)$$

由此可以看出，无迹 Kalman 滤波在处理非线性滤波时并不需要在估计点处做 Taylor 级数展开，然后再进行前 n 阶近似，而是在估计点附近进行 UT 变换，使获得的 Sigma 点集的均值和协方差与原统计特性匹配，再直接对这些 Sigma 点集进

行非线性映射，以近似得到状态概率密度函数。这种近似其实质是一种统计近似而非解。

5.2　无迹 Kalman 滤波在单观测站目标跟踪中的应用

5.2.1　原理介绍

假定目标做匀速直线运动，在单个观测站对目标进行观测的前提下，再假设目标的初始状态已知。由 4.3 节内容可知，目标的运动方程可以写成如下形式：

$$\boldsymbol{X}(k+1)=\boldsymbol{\Phi}\boldsymbol{X}(k)+\boldsymbol{\Gamma}\boldsymbol{W}(k) \tag{5.4}$$

$$\boldsymbol{Z}(k)=\sqrt{(x(k)-x_0)^2+(y(k)-y_0)^2}+\boldsymbol{V}(k) \tag{5.5}$$

式中，

$$\boldsymbol{\Phi}=\begin{bmatrix}1 & T & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & T\\ 0 & 0 & 0 & 1\end{bmatrix},\quad \boldsymbol{\Gamma}=\begin{bmatrix}T^2/2 & 0\\ T & 0\\ 0 & T^2/2\\ 0 & T\end{bmatrix}$$

设采样时间间隔为 T=1s，运行的总时间 N=60s，则过程驱动矩阵和噪声驱动矩阵是常数矩阵，即

$$\boldsymbol{\Phi}=\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\end{bmatrix},\quad \boldsymbol{\Gamma}=\begin{bmatrix}0.5 & 0\\ 1 & 0\\ 0 & 0.5\\ 0 & 1\end{bmatrix}$$

$\boldsymbol{W}(k)$ 的均方差为 $\boldsymbol{Q}=\sigma_{\mathrm{w}}$*diag([1,1])，$\sigma_{\mathrm{w}}$ 为一个可调节的参数，σ_{w}<<1。$\boldsymbol{V}(k)$ 的均方差为 R=5。同时设置 UT 变换中的相关系数，$\alpha=0.01$，$\kappa=0$，$\beta=2$，维数 n=9。雷达所处的位置可以是任意的，在这里给定其位置为（200, 300）。图 5.2 是基于 UKF 算法的跟踪轨迹图。

假设目标运动各时刻的真实状态信息为

$$\boldsymbol{X}_{\mathrm{real}}(k)=[x_{\mathrm{real}}(k),\dot{x}_{\mathrm{real}}(k),y_{\mathrm{real}}(k),\dot{y}_{\mathrm{real}}(k)]^{\mathrm{T}}$$

而利用 UKF 滤波算法得到的目标状态为

$$\boldsymbol{X}_{\mathrm{UKF}}(k)=[x_{\mathrm{UKF}}(k),\dot{x}_{\mathrm{UKF}}(k),y_{\mathrm{UKF}}(k),\dot{y}_{\mathrm{UKF}}(k)]^{\mathrm{T}}$$

定义均方根误差（RMSE）：

$$\mathrm{RMSE}(k)=\sqrt{(x_{\mathrm{UKF}}(k)-x_{\mathrm{real}}(k))^2+(y_{\mathrm{UKF}}(k)-y_{\mathrm{real}}(k))^2} \tag{5.6}$$

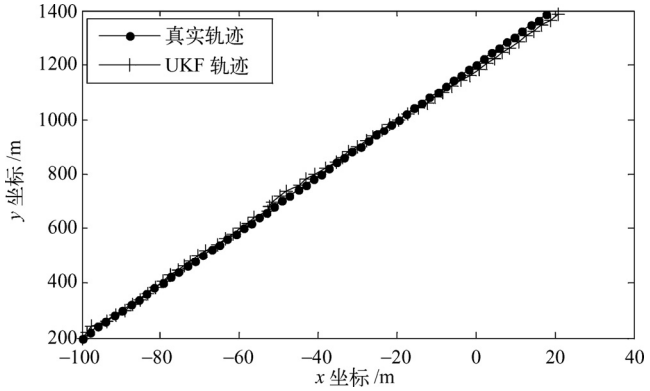


图 5.2　基于 UKF 算法的跟踪轨迹图

那么根据此定义，目标运行各时刻，利用 UKF 计算得到位置与目标真实位置的偏差，其结果如图 5.3 所示。

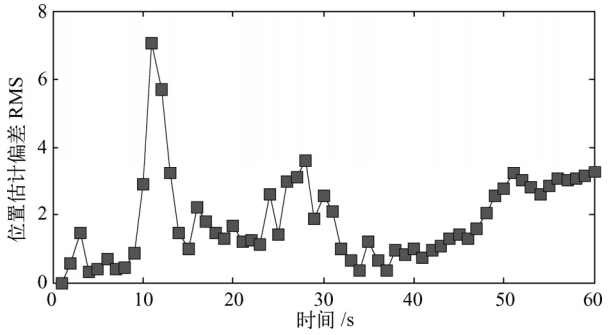


图 5.3　UKF 算法误差曲线图

5.2.2　仿真程序

```
%%%
```

```
X(:,1)=[-100,2,200,20];            % 目标初始位置、速度
Z=zeros(1,N);                      % 传感器对位置的观测
delta_w=1e-3;                      % 如果增大这个参数，目标真实轨迹就是曲线了
Q=delta_w*diag([0.5,1]) ;          % 过程噪声均值
G=[T^2/2,0;T,0;0,T^2/2;0,T];               % 过程噪声驱动矩阵
R=5;                                       % 观测噪声方差
F=[1,T,0,0;0,1,0,0;0,0,1,T;0,0,0,1];       % 状态转移矩阵
x0=200;                                    % 观测站的位置，可以设为其他值
y0=300;
Xstation=[x0,y0];                          % 雷达站的位置
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v=sqrtm(R)*randn(1,N);
for t=2:N
    X(:,t)=F*X(:,t-1)+G*sqrtm(Q)*randn(2,1);  %目标真实轨迹
end
for t=1:N
    Z(t)=Dist(X(:,t),Xstation)+v(t);          %对目标观测
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UKF 滤波, UT 变换
L=4;
alpha=1;
kalpha=0;
belta=2;
ramda=3-L;
for j=1:2*L+1
    Wm(j)=1/(2*(L+ramda));
    Wc(j)=1/(2*(L+ramda));
end
Wm(1)=ramda/(L+ramda);
Wc(1)=ramda/(L+ramda)+1-alpha^2+belta;     % 权值计算
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Xukf=zeros(4,N);
Xukf(:,1)=X(:,1);                          % 无迹 Kalman 滤波状态初始化
P0=eye(4);                                 % 协方差阵初始化
for t=2:N
    xestimate= Xukf(:,t-1);
    P=P0;
    %第一步：获得一组 Sigma 点集
    cho=(chol(P*(L+ramda)))';
    for k=1:L
        xgamaP1(:,k)=xestimate+cho(:,k);
        xgamaP2(:,k)=xestimate-cho(:,k);
    end
    Xsigma=[xestimate,xgamaP1,xgamaP2];    %Sigma 点集
    % 第二步：对 Sigma 点集进行一步预测
    Xsigmapre=F*Xsigma;
```

```
    %第三步：利用第二步的结果计算均值和协方差
    Xpred=zeros(4,1);                    % 均值
    for k=1:2*L+1
        Xpred=Xpred+Wm(k)*Xsigmapre(:,k);
    end
    Ppred=zeros(4,4);                    % 协方差阵预测
    for k=1:2*L+1
Ppred=Ppred+Wc(k)*(Xsigmapre(:,k)-Xpred)*(Xsigmapre(:,k)-Xpred)';
    end
    Ppred=Ppred+G*Q*G';

    % 第四步：根据预测值，再一次使用 UT 变换，得到新的 sigma 点集
    chor=(chol((L+ramda)*Ppred))';
    for k=1:L
        XaugsigmaP1(:,k)=Xpred+chor(:,k);
        XaugsigmaP2(:,k)=Xpred-chor(:,k);
    end
    Xaugsigma=[Xpred XaugsigmaP1 XaugsigmaP2];

    % 第五步：观测预测
    for k=1:2*L+1                        % 观测预测
        Zsigmapre(1,k)=hfun(Xaugsigma(:,k),Xstation);
    end

    % 第六步：计算观测预测均值和协方差
    Zpred=0;                             % 观测预测的均值
    for k=1:2*L+1
        Zpred=Zpred+Wm(k)*Zsigmapre(1,k);
    end
    Pzz=0;
    for k=1:2*L+1
Pzz=Pzz+Wc(k)*(Zsigmapre(1,k)-Zpred)*(Zsigmapre(1,k)-Zpred)';
    end
    Pzz=Pzz+R;                           % 得到协方差 Pzz

    Pxz=zeros(4,1);
    for k=1:2*L+1
Pxz=Pxz+Wc(k)*(Xaugsigma(:,k)-Xpred)*(Zsigmapre(1,k)-Zpred)';
    end
    % 第七步：计算 Kalman 增益
    K=Pxz*inv(Pzz);                      % Kalman 增益
    %第八步：状态和方差更新
    xestimate=Xpred+K*(Z(t)-Zpred);% 状态更新
    P=Ppred-K*Pzz*K';                    % 方差更新
    P0=P;
    Xukf(:,t)=xestimate;
end
```

```
% 误差分析
for i=1:N
    Err_KalmanFilter(i)=Dist(X(:,i),Xukf(:,i));          % 滤波后的误差
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 画图
figure
hold on;box on;
plot(X(1,:),X(3,:),'-k.');                               % 真实轨迹
plot(Xukf(1,:),Xukf(3,:),'-r+');                         % 无迹 Kalman 滤波轨迹
legend('真实轨迹','UKF 轨迹')
figure
hold on; box on;
plot(Err_KalmanFilter,'-ks','MarkerFace','r')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 子函数:求两点间的距离
function d=Dist(X1,X2)
if length(X2)<=2
    d=sqrt( (X1(1)-X2(1))^2 + (X1(3)-X2(2))^2 );
else
    d=sqrt( (X1(1)-X2(1))^2 + (X1(3)-X2(3))^2 );
end
% 观测子函数：观测距离
function [y]=hfun(x,xx)
y=sqrt((x(1)-xx(1))^2+(x(3)-xx(2))^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.3　UKF 在匀加速度直线运动目标跟踪中的应用

5.3.1　原理介绍

匀加速直线运动模型的建模过程与匀速直线运动模型类似，可以参考 4.3 节。考虑一个在二维平面 x-y 内运动的质点 M，其在某一时刻 k 的位置、速度和加速度可用矢量 $\boldsymbol{X}(k)=[x_k,y_k,\dot{x}_k,\dot{y}_k,\ddot{x}_k,\ddot{y}_k]^{\mathrm{T}}$ 表示。假设 M 在水平方向上（x 轴方向）作近似匀加速直线运动，垂直方向上（y 轴方向）亦作近似匀加速直线运动。两方向上运动都具有加性系统噪声 $\boldsymbol{W}(k)$，则在笛卡儿坐标系下该质点的运动状态方程为

$$\boldsymbol{X}(k+1)=\boldsymbol{\Phi}\boldsymbol{X}(k)+\boldsymbol{W}(k) \tag{5.7}$$

式中，

$$\boldsymbol{\Phi}=\begin{bmatrix}1&0&T&0&\frac{T^2}{2}&0\\0&1&0&T&0&\frac{T^2}{2}\\0&0&1&0&T&0\\0&0&0&1&0&T\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}$$

假设坐标位置为（x_0, y_0）的雷达对质点 M 进行跟踪，则可以得到雷达和质点 M 之间的距离 r_k 和质点 M 相对于雷达的角度 φ_k，实际测量中雷达具有加性测量噪声 $\boldsymbol{V}(k)$，在以雷达为中心的坐标系下，观测方程为

$$\begin{aligned}\boldsymbol{Z}(k)&=\boldsymbol{h}(\boldsymbol{X}(k))+\boldsymbol{V}(k)=\begin{bmatrix}r(k)+\boldsymbol{V}_r(k)\\\varphi(k)+\boldsymbol{V}_\varphi(k)\end{bmatrix}\\&=\begin{bmatrix}\sqrt{(x(k)-x_0)^2+(y(k)-y_0)^2}+\boldsymbol{V}_r(k)\\\arctan\left(\dfrac{y(k)-y_0}{x(k)-x_0}\right)+\boldsymbol{V}_\varphi(k)\end{bmatrix}\end{aligned}\tag{5.8}$$

在笛卡儿坐标系下，该模型的状态方程是线性的，而观测方程是非线性的。读者可以进一步将该系统扩展到三维空间 x–y–z 坐标系下，那么目标的状态则是 9 维信息，更符合实际应用情况。

在仿真中假设系统噪声 $\boldsymbol{W}(k)$ 具有协方差阵 $\boldsymbol{Q}_k$，$\boldsymbol{V}(k)$ 具有协方差阵 $\boldsymbol{R}_k$，分别如下。

$$\boldsymbol{Q}_k=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&0.1^2&0&0&0\\0&0&0&0.1^2&0&0\\0&0&0&0&0.01^2&0\\0&0&0&0&0&0.01^2\end{bmatrix},\quad \boldsymbol{R}_k=\begin{bmatrix}5^2&0\\0&0.01^2\end{bmatrix}$$

$\boldsymbol{W}$、$\boldsymbol{V}$ 二者不相关，观测次数 N=50，采样时间为 T=0.5s。初始状态 $\boldsymbol{X}(0)=[1000,5000,10,50,2,-4]^{\mathrm{T}}$，则生成的运动轨迹如图 5.4 所示，跟踪位置误差如图 5.5 所示。

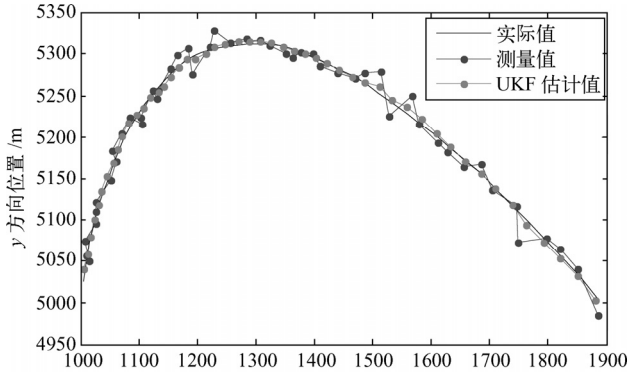


图 5.4　运动轨迹图

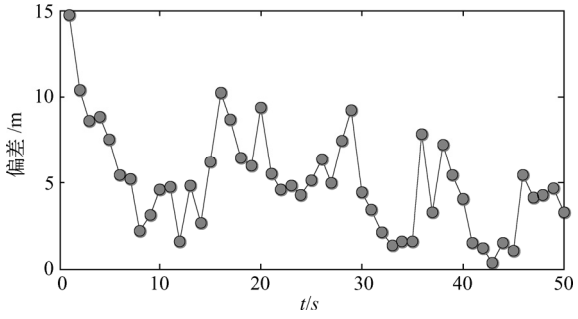


图 5.5　跟踪误差图

5.3.2　仿真程序

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   功能说明：  UKF 在目标跟踪中的应用
%   参数说明：  1．状态 6 维，x 方向的位置、速度、加速度；
%                   y 方向的位置、速度、加速度；
%               2．观测信息为距离和角度；
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function ukf_for_track_6_div_system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=6;                % 状态位数
t=0.5;              % 采样时间
Q=[1 0 0 0 0 0;
    0 1 0 0 0 0;
    0 0 0.01 0 0 0;
    0 0 0 0.01 0 0;
    0 0 0 0 0.0001 0;
    0 0 0 0 0 0.0001];          %过程噪声协方差阵
R = [100 0;
    0 0.001^2];                 %量测噪声协方差阵
```

```
% 状态方程
f=@(x)[x(1)+t*x(3)+0.5*t^2*x(5);x(2)+t*x(4)+0.5*t^2*x(6);...
    x(3)+t*x(5);x(4)+t*x(6);x(5);x(6)];
%x1 为 X 轴位置，x2 为 Y 轴位置，x3、x4 分别是 X、
%Y 轴的速度，x5、x6 为 X、Y 两方向的加速度
% 观测方程
h=@(x)[sqrt(x(1)^2+x(2)^2);atan(x(2)/x(1))];
s=[1000;5000;10;50;2; -4];
x0=s+sqrtm(Q)*randn(n,1);          % 初始化状态
P0 =[100 0 0 0 0 0;
    0 100 0 0 0 0;
    0 0 1 0 0 0;
    0 0 0 1 0 0;
    0 0 0 0 0.1 0;
    0 0 0 0 0 0.1];                % 初始化协方差
N=50;                              % 总仿真时间步数，即总时间
Xukf = zeros(n,N);                 % UKF 滤波状态初始化
X = zeros(n,N);                    % 真实状态
Z = zeros(2,N);                    % 测量值
for i=1:N
    X(:,i)= f(s)+sqrtm(Q)*randn(6,1);          % 模拟，产生目标运动真实轨迹
    s = X(:,i);
end
ux=x0;                                         % ux 为中间变量
for k=1:N
    Z(:,k)= h(X(:,k)) + sqrtm(R)*randn(2,1);   % 测量值     % 保存观测
    [Xukf(:,k),P0] = ukf(f,ux,P0,h,Z(:,k),Q,R);% 调用 ukf 滤波算法
    ux=Xukf(:,k);
end
% 跟踪误差分析
% 这里只分析位置误差，速度、加速度误差分析在此略，读者可以自己尝试
for k=1:N
    RMS(k)=sqrt( (X(1,k) -Xukf(1,k))^2+(X(2,k) -Xukf(2,k))^2 );
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 画图，轨迹图
figure
t=1:N;
hold on;box on;
plot( X(1,t),X(2,t), 'k-')
plot(Z(1,t).*cos(Z(2,t)),Z(1,t).*sin(Z(2,t)),'-b.')
plot(Xukf(1,t),Xukf(2,t),'-r.')
legend('实际值','测量值','UKF 估计值');
xlabel('x 方向位置/m')
ylabel('y 方向位置/m')
% 误差分析图
```

```
figure
box on;
plot(RMS,'-ko','MarkerFace','r')
xlabel('t/s')
ylabel('偏差/m')
%title('跟踪位置偏差')
%%%
```

```
% 输出：Xmeans 为均值，
function [Xmeans,Xsigma_pre,P,Xdiv]=ut(fun,Xsigma,Wm,Wc,n,COV)
LL=size(Xsigma,2);% 得到 Xsigma 样本个数
Xmeans=zeros(n,1);%均值
Xsigma_pre=zeros(n,LL);
for k=1:LL
    Xsigma_pre(:,k)=fun(Xsigma(:,k));          % 一步预测
    Xmeans=Xmeans+Wm(k)*Xsigma_pre(:,k);
end
% Xmeans(:,ones(1,LL))将 Xmeans 扩展成 n*LL 矩阵，每一列都相等
Xdiv=Xsigma_pre-Xmeans(:,ones(1,LL));          % 预测减去均值
P=Xdiv*diag(Wc)*Xdiv'+COV;                     % 协方差
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 产生 Sigma 点集函数
function Xset=sigmas(X,P,c)
A = c*chol(P)';% Cholesky 分解
Y = X(:,ones(1,numel(X)));
Xset = [X Y+A Y-A];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.4 UKF 与 EKF 算法的应用比较

为了对比 UKF 和 EKF 两种算法的滤波效果，我们选用下面的系统进行仿真分析。这里选用一维非线性系统，状态方程为

$$X(k)=0.5X(k-1)+\frac{2.5X(k-1)}{1+X(k-1)^2}+8\cos(1.2k)+W(k)$$

观测方程为

$$Z(k)=\frac{X(k)^2}{20}+V(k)$$

该系统中，过程噪声 $W(k)$，它的方差为 Q，观测方程中的噪声为 $V(k)$，其方差为 R。

在 EKF 算法中，对状态方程和观测方程线性化求雅克比矩阵的方法如下。

$$F=\frac{\partial f}{\partial X}=0.5+\frac{2.5(1+X(k-1)^2)-5X(k-1)^2}{(1+X(k-1)^2)^2}$$

$$H=\frac{\partial h}{\partial X}=\frac{X(k)}{10}$$

而在 UKF 的 UT 变换中，由于系统维数 L=1，取常数 $\alpha=1$，$\kappa=0$，$\beta=2$，

$\lambda = 3 - L$。

k=1～N，N=50；过程噪声 Q=10，观测噪声 R=1，仿真该系统，得到状态估计的结果如图 5.6 所示。

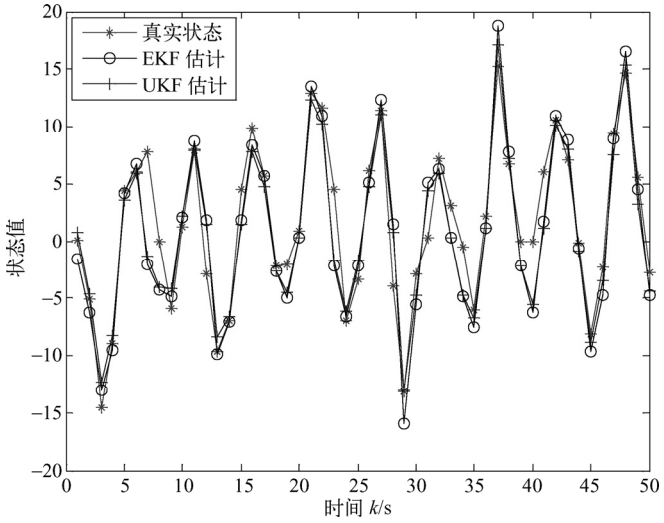


图 5.6　EKF 和 UKF 两种状态估计结果对比

为了更直观显示状态估计的准确性，我们将每个时刻的估计值与真实值做差，得到绝对值偏差，即

$$\mathrm{RMS}(k) = \left| X_{\text{estimate}} - X_{\text{real}} \right|$$

得到 EKF 和 UKF 各个时刻的估计偏差结果如图 5.7 所示，从图中可以看出，k=1～N 时，并不是每一次 UKF 估计的结果都比 EKF 估计的更准确，但是整体上来看，UKF 的估计偏差会比 EKF 小。

为了衡量误差的整体水平，我们做了多次仿真实验，每次实验误差的平均值定义为

$$\mathrm{RMSE} = \frac{1}{N}\sum_{k=1}^{N} \mathrm{RMS}(k)$$

从表 5.1 中可以看出，对 N 个时间序列的估计求平均值，那么 UKF 的误差均值大多数情况是比 EKF 小的，这表明状态估计的准确性上 UKF 优于 EKF，但不是绝对的，比如表 5.1 中第 3 次试验，EKF 的均值较小，也就是说，UKF 从概率统计意义上优于 EKF。

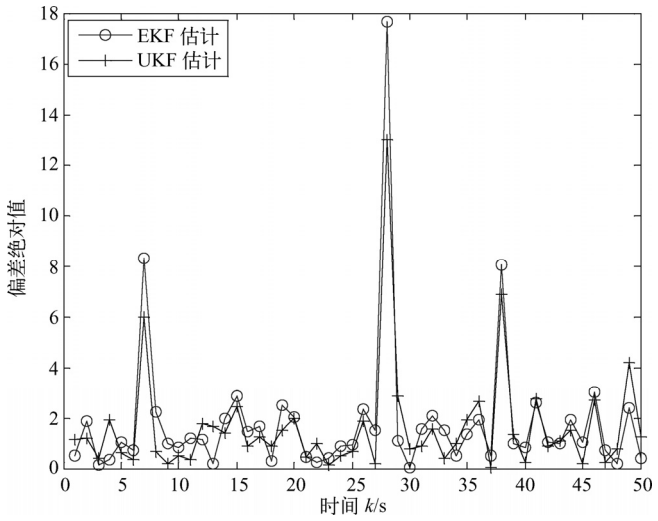


图 5.7　EKF 和 UKF 各个时刻的估计偏差对比

表 5.1　多次试验的平均值

	1	2	3	4	5	6	7	8	……
EKF	2.5738	1.5331	1.5093	1.7568	1.6331	2.0469	1.6692	1.8235	……
UKF	2.1106	1.3132	1.5310	1.6758	1.4332	1.8934	1.5931	1.6121	……

由该仿真分析可以看出：在相同条件下，UKF 具有较 EKF 更高的滤波精度。对 RMSE 的均值和方差的计算则更有力地证明了 UKF 算法相对 EKF 算法的优越性。

UKF 算法以贝叶斯理论和 UT 变换为基础，在很大程度上克服了算法的线性化误差，具有广泛的应用前景。UKF 算法的缺点在于其参数的选择问题尚没有得到完全解决，而且其滤波效果与 EKF 算法一样也受到滤波初值的影响。

第 6 章　交互多模型 Kalman 滤波

在 Kalman 滤波算法中用到了状态转移方程和观测方程，被估计量随时间变化，它是一种动态估计。在目标跟踪中，不必知道目标的运动模型就能够实时地修正目标的状态参量（位置、速度等），具有较好的适应性。但是当目标实施机动时（突然转弯或加、减速等），仅采用基本的 Kalman 滤波算法往往得不到理想的结果。这时需要采用自适应算法。交互多模型（IMM）是一种软切换算法，最初由 H.A. PBlom 在 1984 年提出，目前在机动目标跟踪领域得到了广泛的应用。IMM 算法使用两个或更多的模型来描述工作过程中可能的状态，最后通过有效的加权融合进行系统状态估计，很好地克服了单模型估计误差较大的问题。

6.1　交互多模型 Kalman 滤波原理

IMM 算法采用多个 Kalman 滤波器进行并行处理。每个滤波器对应不同的状态空间模型，不同的状态空间模型描述不同的目标运行模式，所以每个滤波器对目标状态的估计结果不同。IMM 算法的基本思想是在每一时刻，假设某个模型在现在时刻有效的条件下，通过混合前一时刻所有滤波器的状态估计值来获得与这个特定模型匹配的滤波器的初始条件，然后对每个模型并行实现正规滤波（预测和修正）步骤；最后，以模型匹配似然函数为基础更新模型概率，并组合所有滤波器修正后的状态估计值（加权和）以得到状态估计。因此 IMM 算法的估计结果是对不同模型所得估计的混合，而不是仅仅在每一个时刻选择完全正确的模型来估计。下面介绍 IMM 算法的一般步骤。

假定目标有 r 种运动状态，对应有 r 个运动模型（即 r 个状态转移方程），设第 j 个模型表示的目标状态方程为

$$\boldsymbol{X}_j(k+1)=\boldsymbol{\Phi}_j(k)\boldsymbol{X}_j(k)+\boldsymbol{G}_j(k)\boldsymbol{W}_j(k) \tag{6.1}$$

量测方程为

$$\boldsymbol{Z}(k)=\boldsymbol{H}(k)\boldsymbol{X}(k)+\boldsymbol{V}(k) \tag{6.2}$$

式中，$\boldsymbol{W}_j(k)$是均值为零，协方差矩阵为$\boldsymbol{Q}_j$的白噪声序列。各模型之间的转移由马尔可夫概率转移矩阵确定，其中的元素p_{ij}表示目标由第i个运动模型转移到第j个运动模型的概率，概率转移矩阵如下。

$$\boldsymbol{P}=\begin{bmatrix} p_{11} & \cdots & p_{1r} \\ \cdots & \cdots & \cdots \\ p_{r1} & \cdots & p_{rr} \end{bmatrix} \tag{6.3}$$

IMM 算法是以递推方式进行的，每次递推主要分为以下四个步骤。

步骤 1：输入交互（模型j）

由目标的状态估计$\hat{\boldsymbol{X}}_i(k-1|k-1)$与上一步中每个滤波器的模型概率$\boldsymbol{\mu}_j(k-1)$得到混合估计$\hat{\boldsymbol{X}}_{0j}(k-1|k-1)$和协方差$\boldsymbol{P}_{0j}(k-1|k-1)$，将混合估计作为当前循环的初始状态。具体的参数计算如下：

模型j的预测概率（归一化常数）为

$$\overline{\boldsymbol{c}}_j=\sum_{i=1}^{r}p_{ij}\boldsymbol{\mu}_i(k-1) \tag{6.4}$$

模型i到模型j的混合概率

$$\boldsymbol{\mu}_{ij}(k-1|k-1)=\sum_{i=1}^{r}p_{ij}\boldsymbol{\mu}_i(k-1)/\overline{\boldsymbol{c}}_j \tag{6.5}$$

模型j的混合状态估计

$$\hat{\boldsymbol{X}}_{0j}(k-1|k-1)=\sum_{i=1}^{r}\hat{\boldsymbol{X}}_i(k-1|k-1)\boldsymbol{\mu}_{ij}(k-1|k-1) \tag{6.6}$$

模型j的混合协方差估计

$$\begin{aligned}\boldsymbol{P}_{0j}(k-1|k-1)=\sum_{i=1}^{r}\boldsymbol{\mu}_{ij}(k-1|k-1)\{&\boldsymbol{P}_i(k-1|k-1)\\&+[\hat{\boldsymbol{X}}_i(k-1|k-1)-\hat{\boldsymbol{X}}_{0j}(k-1|k-1)]\\&\cdot[\hat{\boldsymbol{X}}_i(k-1|k-1)-\hat{\boldsymbol{X}}_{0j}(k-1|k-1)]^{\mathrm{T}}\}\end{aligned} \tag{6.7}$$

式中，p_{ij}模型i到模型j的转移概率；$\boldsymbol{\mu}_j(k-1)$为模型j在$k-1$时刻的概率。

步骤 2：Kalman 滤波（模型 j）

以 $\hat{\boldsymbol{X}}_{0j}(k-1|k-1)$、$\boldsymbol{P}_{0j}(k-1|k-1)$ 及 $\boldsymbol{Z}(k)$ 作为输入进行 Kalman 滤波，来更新预测状态 $\hat{\boldsymbol{X}}_j(k|k)$ 和滤波协方差 $\boldsymbol{P}_j(k|k)$。

预测

$$\hat{\boldsymbol{X}}_j(k|k-1)=\boldsymbol{\Phi}_j(k-1)\hat{\boldsymbol{X}}_{0j}(k-1|k-1) \tag{6.8}$$

预测误差协方差

$$\boldsymbol{P}_j(k|k-1)=\boldsymbol{\Phi}_j\boldsymbol{P}_{0j}(k-1|k-1)\boldsymbol{\Phi}_j{}^{\mathrm{T}}+\boldsymbol{G}_j\boldsymbol{Q}_j\boldsymbol{G}_j{}^{\mathrm{T}} \tag{6.9}$$

Kalman 增益

$$\boldsymbol{K}_j(k)=\boldsymbol{P}_j(k|k-1)\boldsymbol{H}^{\mathrm{T}}[\boldsymbol{H}\boldsymbol{P}_j(k|k-1)\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R}]^{-1} \tag{6.10}$$

滤波

$$\hat{\boldsymbol{X}}_j(k|k)=\hat{\boldsymbol{X}}_j(k|k-1)+\boldsymbol{K}_j(k)[\boldsymbol{Z}(k)-\boldsymbol{H}(k)\boldsymbol{X}_j(k|k-1)] \tag{6.11}$$

滤波协方差

$$\boldsymbol{P}_j(k|k)=[\boldsymbol{I}-\boldsymbol{K}_j(k)\boldsymbol{H}(k)]\boldsymbol{P}_j(k|k-1) \tag{6.12}$$

步骤 3：模型概率更新

采用似然函数来更新模型概率 $\mu_j(k)$，模型 j 的似然函数为

$$\boldsymbol{\Lambda}_j(k)=\frac{1}{(2\pi)^{n/2}|\boldsymbol{S}_j(k)|^{1/2}}\exp\left\{-\frac{1}{2}\boldsymbol{v}_j{}^{T}\boldsymbol{S}_j^{-1}(k)\boldsymbol{v}_j\right\} \tag{6.13}$$

式中，

$$\boldsymbol{v}_j(k)=\boldsymbol{Z}(k)-\boldsymbol{H}(k)\hat{\boldsymbol{X}}_j(k|k-1)$$

$$\boldsymbol{S}_j(k)=\boldsymbol{H}(k)\boldsymbol{P}_j(k|k-1)\boldsymbol{H}(k)^{\mathrm{T}}+\boldsymbol{R}(k)$$

则模型 j 的概率为

$$\boldsymbol{\mu}_j(k)=\boldsymbol{\Lambda}_j(k)\overline{c}_j/c \tag{6.14}$$

式中，c 为归一化常数，且 $c=\sum_{j=1}^{r}\boldsymbol{\Lambda}_j(k)\overline{c}_j$。

步骤 4：输出交互

基于模型概率，对每个滤波器的估计结果加权合并，得到总的状态估计 $\hat{\boldsymbol{X}}(k|k)$ 和总的协方差估计 $\boldsymbol{P}(k|k)$。

总的状态估计

$$\hat{\boldsymbol{X}}(k|k)=\sum_{j=1}^{r}\hat{\boldsymbol{X}}_j(k|k)\boldsymbol{\mu}_j(k) \tag{6.15}$$

总的协方差估计

$$\boldsymbol{P}(k|k)=\sum_{j=1}^{r}\boldsymbol{\mu}_j(k)\{\boldsymbol{P}_j(k|k)+[\hat{\boldsymbol{X}}_j(k|k)-\hat{\boldsymbol{X}}(k|k)]\cdot[\hat{\boldsymbol{X}}_j(k|k)-\hat{\boldsymbol{X}}(k|k)]^{\mathrm{T}}\} \tag{6.16}$$

所以，滤波器的总输出是多个滤波器估计结果的加权平均值。权重即为该时刻模型正确描述目标运动的概率，简称为模型概率。

选取滤波器的目标运动模型，可以从下面 3 个方面考虑：

（1）选择一定个数的 IMM 滤波器，包括较为精确的模型和较为粗糙的模型。IMM 滤波算法不仅描述了目标的连续运动状态，而且描述了目标的机动性。

（2）马尔可夫链状态转移概率的选取对 IMM 滤波器的性能有较大影响。马尔可夫链状态转移概率矩阵实际上相当于模型状态方程的状态转移矩阵，它将直接影响模型误差和模型概率估计的准确性。一般情况下，当马尔可夫链状态转移概率呈现一定程度的模型性时，IMM 滤波器能够更稳健地描述目标运动。

（3）IMM 滤波算法具有模块化的特性。当对目标的运动规律较为清楚时，滤波器可以选择能够比较精确地描述目标运动的模型。当无法预料目标的运动规律时，就应该选择更一般的模型，即该模型应具有较强的鲁棒性。

6.2 交互多模型 Kalman 滤波在目标跟踪中的应用

6.2.1 问题描述

假定有一雷达对平面上运动的目标进行观测。目标在$t=0\sim40$s 沿 y 轴作匀速直线运动，运动速度为−15m/s，目标的起始点为（2000m,10000m）；在$t=400\sim600$s 向 x 轴方向做 90° 的慢转弯，加速度为$u_x=u_y=0.075\mathrm{m/s}$，完成慢转弯后加速度将降为零；从$t=610$s 开始做 90° 的快转弯，加速度为 0.3m/s^2；在 660s 结束转弯，加速度降至零。雷达扫描周期$T=2$s，x 和 y 独立地进行观测，观测噪声的标准差

均为 100m。试建立雷达对目标的跟踪算法，并进行仿真分析，给出仿真分析结果，画出目标的真实轨迹、目标的观测和滤波曲线。

6.2.2　IMM 滤波器设计

对于上述问题，我们采用三个模型：第一个模型是非机动模型，假定它的系统扰动噪声方差为零，即不考虑 $\boldsymbol{W}$ 的影响；第二、三个模型为机动模型，假定第二个模型的系统扰动噪声方差为 $\boldsymbol{Q}=0.001\boldsymbol{I}_{2\times2}$，第三个模型的系统扰动噪声方差为 $\boldsymbol{Q}=0.0144\boldsymbol{I}_{2\times2}$。控制模型转换的马尔可夫链的转移概率矩阵为

$$\boldsymbol{P}=\begin{bmatrix}0.95 & 0.025 & 0.025\\ 0.025 & 0.95 & 0.025\\ 0.025 & 0.025 & 0.95\end{bmatrix}$$

在跟踪的初始阶段，首先采用常规 Kalman 滤波（非机动模型）进行跟踪，从第 20 次采样开始，采用三个模型的 IMM 算法。设定各模型在此时刻的概率分别为 $\mu_1=0.8$，$\mu_2=0.1$，$\mu_3=0.1$。

各模型参数定义如下。

$$\boldsymbol{X}_1=\begin{bmatrix}x\\ \dot{x}\\ y\\ \dot{y}\\ 0\\ 0\end{bmatrix},\quad \boldsymbol{X}_2=\boldsymbol{X}_3=\begin{bmatrix}x\\ \dot{x}\\ y\\ \dot{y}\\ \ddot{x}\\ \ddot{y}\end{bmatrix},\quad \boldsymbol{G}_1=\begin{bmatrix}T^2/2 & 0\\ T & 0\\ 0 & T^2/2\\ 0 & T\\ 0 & 0\\ 0 & 0\end{bmatrix},\quad \boldsymbol{G}_2=\boldsymbol{G}_3=\begin{bmatrix}T^2/4 & 0\\ T/2 & 0\\ 0 & T^2/4\\ 0 & T\\ 1 & 0\\ 0 & 1\end{bmatrix}$$

$$\boldsymbol{\Phi}_1=\begin{bmatrix}1 & T & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & T & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix},\quad \boldsymbol{\Phi}_2=\boldsymbol{\Phi}_3=\begin{bmatrix}1 & T & 0 & 0 & T^2/2 & 0\\ 0 & 1 & 0 & 0 & T & 0\\ 0 & 0 & 1 & T & 0 & T^2/2\\ 0 & 0 & 0 & 1 & 0 & T\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}$$

采用两点起始法（航迹起始算法之一），求得初始状态为

$$\hat{\boldsymbol{X}}(2\,|\,2)=\begin{bmatrix} z_x(2) & \dfrac{z_x(2)-z_x(1)}{T} & z_y(2) & \dfrac{z_y(2)-z_y(1)}{T} & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{P}(2\,|\,2)=\begin{bmatrix} \sigma_x^2 & \sigma_x^2/T & 0 & 0 & 0 & 0 \\ \sigma_x^2/T & 2\sigma_x^2/T^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_y^2 & \sigma_y^2/T & 0 & 0 \\ 0 & 0 & \sigma_y^2/T & 2\sigma_y^2/T^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

定义滤波误差的均值

$$\overline{\boldsymbol{e}_x(k)}=\frac{1}{M}\sum_{i=1}^{M}[\boldsymbol{x}_i(k)-\hat{\boldsymbol{x}}_i(k\,|\,k)]$$

定义滤波误差的标准差

$$\sigma_{\hat{x}}=\sqrt{\frac{1}{M}\sum_{i=1}^{M}[\boldsymbol{x}_i(k)-\hat{\boldsymbol{x}}_i(k/k)]^2-\left[\overline{\boldsymbol{e}_x(k)}\right]^2}$$

式中，M 为蒙特卡洛模拟次数 $k=1,2,\cdots,N$，N 为采样次数。

6.2.3 仿真分析

设定情景目标的真实轨迹、观测轨迹及一次滤波结果曲线如图 6.1 所示。采用蒙特卡洛方法仿真 50 次，得到滤波误差的均值曲线和标准差曲线分别如图 6.2、图 6.3 所示。从图 6.1 中看出，滤波曲线基本上在真实轨迹附近摆动，无论在目标非机动还是机动时，都能较好地跟上目标，达到了比较好的滤波效果。

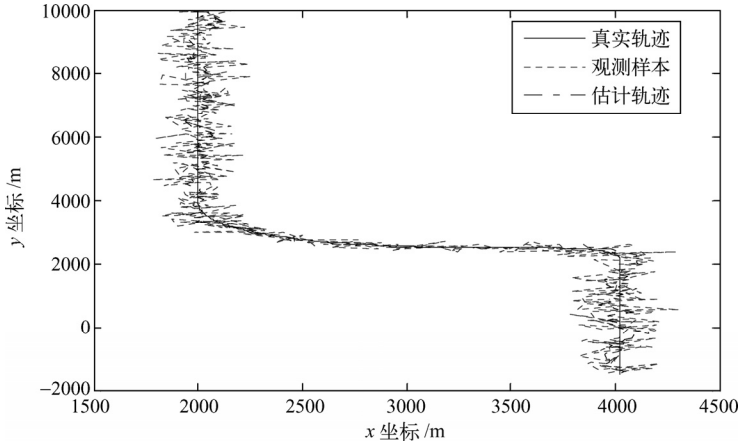


图 6.1　目标的真实轨迹、观测轨迹、估计轨迹（一次滤波曲线）

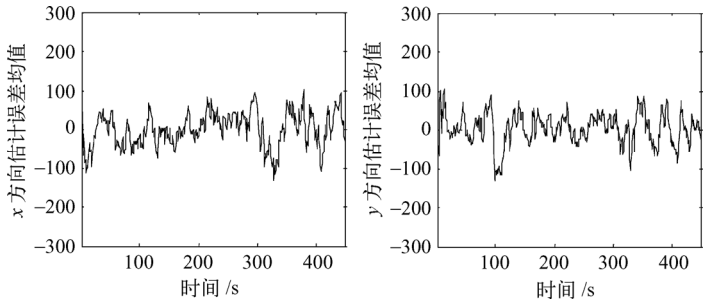


图 6.2　滤波误差的均值曲线

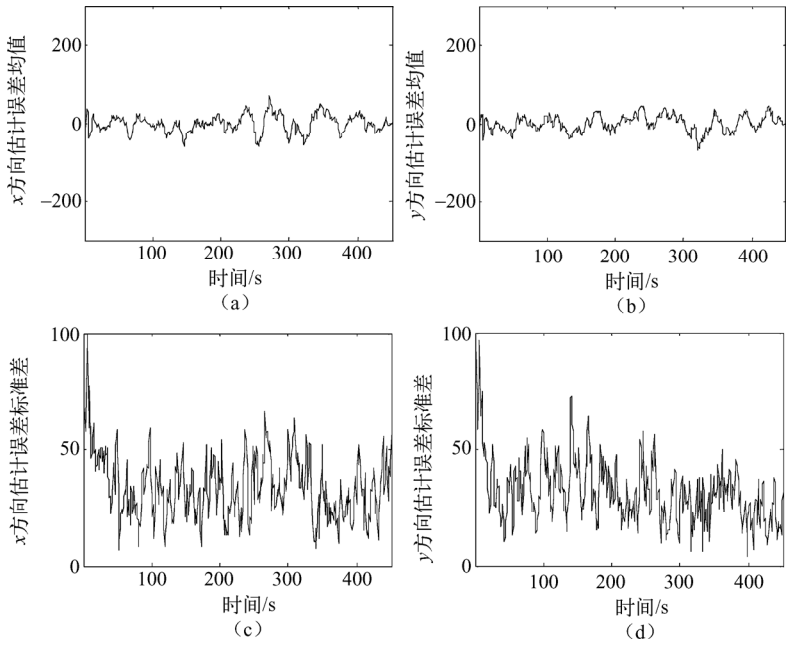


图 6.3　滤波误差的标准差曲线

图 6.1 绘出了在机动情况下的目标滤波曲线，在加入系统扰动噪声后滤波曲线以真实轨迹为中心有很小的扰动，很形象地描述了 Kalman 滤波过程。

图 6.2 表示的是在 x 和 y 方向上的误差均值曲线，在曲线中看出，由于有两次转弯，所以在转弯的前后误差的均值有四次较大的波动，在目标运动的轨迹变为匀速运动时，误差曲线再次在零值附近有很小的波动。

图 6.3 表示的是在 x 和 y 方向上的误差标准差曲线，在图上可以看出第二次转弯较第一次更急，因为在误差标准差曲线上产生了更大的误差标准差。在滤波进入稳定状态后，标准差很小，这说明 IMM 算法受目标机动的影响较小。

从以上的分析可以看出，采用 IMM Kalman 滤波算法可以很快地跟踪机动目

标，并且不会有大的偏差，只是运算量较大。

6.2.4 IMM Kalman 滤波算法 MATLAB 仿真程序

```
%%%
```

```
    y3=y2(N3-N2)  -0.3*((t'-610).^2)/2;
    % 5-匀速直线
    t=662:T:900;
    vy=-0.3*(660-610);
    x4=x3(N4-N3)+0*t';
    y4=y3(N4-N3)+vy*(t'-660);
    % 最终将所有轨迹整合成为一个列向量，即真实轨迹数据,Rx 为 Real-x，Ry 为 Real-y
的简写）
    Rx=[x0;x1;x2;x3;x4];
    Ry=[y0;y1;y2;y3;y4];
    % 对每次蒙特卡洛仿真的滤波估计位置的初始化
    Mt_Est_Px=zeros(M,N);
    Mt_Est_Py=zeros(M,N);
    % 产生观测数据,要仿真 M 次，必须有 M 次的观测数据
    nx=randn(N,M)*Delta;        % 产生观测噪声
    ny=randn(N,M)*Delta;
    Zx=Rx*ones(1,M)+nx;         % 真实值的基础上叠加噪声，即构成计算机模拟的观测值
    Zy=Ry*ones(1,M)+ny;
    %%%%
```

```
            z=[Zx(r,m);Zy(r,m)];
            if r<20
                x_pre=phi*x_est;                %预测
                p_pre=phi*p_est*phi';           %预测误差协方差
                k=p_pre*h'*inv(h*p_pre*h'+q);   %卡尔曼增益
                x_est=x_pre+k*(z-h*x_pre);      %滤波
                p_est=(eye(4) -k*h)*p_pre;      %滤波协方差

                xn(r)=x_est(1);                 %记录采样点滤波数据
                yn(r)=x_est(3);
                Mt_Est_Px(m,r)=x_est(1);        %记录第 m 次仿真滤波估计数据
                Mt_Est_Py(m,r)=x_est(3);
            else
                if r==20
                    X_est=[x_est;0;0];          %扩维
                    P_est=p_est;
                    P_est(6,6)=0;
                    for i=1:3
                        Xn_est{i,1}=X_est;
                        Pn_est{i,1}=P_est;
                    end
                    u=[0.8,0.1,0.1];            %模型概率初始化
                end
                %调用 IMM 算法
                [X_est,P_est,Xn_est,Pn_est,u]=IMM(Xn_est,Pn_est,T,z,Delta,u);
                xn(r)=X_est(1);
                yn(r)=X_est(3);
                Mt_Est_Px(m,r)=X_est(1);
                Mt_Est_Py(m,r)=X_est(3);
            end
        end                                     % 结束一次滤波
    end
    %%%%
```

```
        ey=sum(Ry(r) -Mt_Est_Py(:,r));
        err_x(r)=ex/M;
        err_y(r)=ey/M;
        eqx=sum((Rx(r) -Mt_Est_Px(:,r)).^2);
        eqy=sum((Ry(r) -Mt_Est_Py(:,r)).^2);
        % 估计误差标准差，请对照书中的公式理解
        delta_x(r)=sqrt(abs(eqx/M- (err_x(r)^2)));
        delta_y(r)=sqrt(abs(eqy/M- (err_y(r)^2)));
    end
    %%%%
```

```
% u 为模型概率
function [X_est,P_est,Xn_est,Pn_est,u]=IMM(Xn_est,Pn_est,T,Z,Delta,u)
% 控制模型转换的马尔可夫链的转移概率矩阵
P=[0.95,0.025,0.025;0.025,0.95,0.025;0.025,0.025,0.95];
% 所采用的三个模型参数，模型一为非机动，模型二、三均为机动模型（Q 不同）
% 模型一
PHI{1,1}=[1,T,0,0;0,1,0,0;0,0,1,T;0,0,0,1];
PHI{1,1}(6,6)=0;
PHI{2,1}=[1,T,0,0,T^2/2,0;0,1,0,0,T,0;0,0,1,T,0,T^2/2;
    0,0,0,1,0,T;0,0,0,0,0,1,0;0,0,0,0,0,0,1];          %模型二
PHI{3,1}=PHI{2,1};                                     %模型三
G{1,1}=[T/2,0;1,0;0,T/2;0,1];                          %模型一
G{1,1}(6,2)=0;
G{2,1}=[T^2/4,0;T/2,0;0,T^2/4;0,T/2;1,0;0,1];          %模型二
G{3,1}=G{2,1};                                         %模型三
Q{1,1}=zeros(2);                                       %模型一
Q{2,1}=0.001*eye(2);                                   %模型二
Q{3,1}=0.0114*eye(2);                                  %模型三
H=[1,0,0,0,0,0;0,0,1,0,0,0];
R=eye(2)*Delta^2;                                      %观测噪声协方差阵
mu=zeros(3,3);                                         %混合概率矩阵
c_mean=zeros(1,3);                                     %归一化常数
for i=1:3
    c_mean=c_mean+P(i,:)*u(i);
end
for i=1:3
    mu(i,:)=P(i,:)*u(i)./c_mean;
end
% 输入交互
for j=1:3
    X0{j,1}=zeros(6,1);
    P0{j,1}=zeros(6);
    for i=1:3
        X0{j,1}=X0{j,1}+Xn_est{i,1}*mu(i,j);
    end
    for i=1:3
        P0{j,1}=P0{j,1}+mu(i,j)*( Pn_est{i,1}...
            +(Xn_est{i,1}-X0{j,1})*(Xn_est{i,1}-X0{j,1})');
    end
end
% 模型条件滤波
a=zeros(1,3);
for j=1:3
```

```
        % 观测预测
        X_pre{j,1}=PHI{j,1}*X0{j,1};
        % 协方差预测
        P_pre{j,1}=PHI{j,1}*P0{j,1}*PHI{j,1}'+G{j,1}*Q{j,1}*G{j,1}';
        % 计算卡尔曼增益
        K{j,1}=P_pre{j,1}*H'*inv(H*P_pre{j,1}*H'+R);
        % 状态更新
        Xn_est{j,1}=X_pre{j,1}+K{j,1}*(Z-H*X_pre{j,1});
        % 协方差更新
        Pn_est{j,1}=(eye(6) -K{j,1}*H)*P_pre{j,1};
    end
    % 模型概率更新
    for j=1:3
        v{j,1}=Z-H*X_pre{j,1};                    %新息
        s{j,1}=H*P_pre{j,1}*H'+R;                 %观测协方差矩阵
        n=length(s{j,1})/2;
        a(1,j)=1/((2*pi)^n*sqrt(det(s{j,1})))*exp(-0.5*v{j,1}'...
            *inv(s{j,1})*v{j,1});                 %观测相对于模型 j 的似然函数
    end
    c=sum(a.*c_mean);                             %归一化常数
    u=a.*c_mean./c;                               %模型概率更新
    % 输出交互
    Xn=zeros(6,1);
    Pn=zeros(6);
    for j=1:3
        Xn=Xn+Xn_est{j,1}.*u(j);
    end
    for j=1:3
        Pn=Pn+u(j).*(Pn_est{j,1}+(Xn_est{j,1}-Xn)*(Xn_est{j,1}-Xn)');
    end
    % 返回滤波结果
    X_est=Xn;
    P_est=Pn;
    %%%%
```

第 7 章　Kalman 滤波的 Simulink 仿真

Simulink 是 MATLAB 软件的扩展，是实现动态系统建模和仿真的一个软件包。它与 MATLAB 语言的主要区别是，其与用户交互接口是基于 Windows 的模型化图形输入，使用户可以把更多的精力投入到系统模型的构建，而非语言的编程上。

7.1　Simulink 概述

1990 年，MathWorks 软件公司为 MATLAB 提供了新的系统模型化输入与仿真工具，并命名为 SIMULAB，该工具很快就在工程界获得了广泛的认可，使仿真软件进入了模型化图形组态阶段，但因其名字与当时比较著名的软件 SIMULA 类似，所以在 1992 年正式将该软件更名为 Simulink。

Simulink 的出现为系统仿真与设计带来福音。顾名思义，该软件有两个主要功能：Simu（仿真）和 Link（连接），即该软件可以利用鼠标在模型窗口上绘制出所需要的仿真系统模型，然后利用 Simulink 提供的功能来对系统进行仿真和分析。

Simulink 是 MATLAB 软件的扩展，是实现动态系统建模和仿真的一个软件包。它与 MATLAB 语言的主要区别是，其与用户交互接口是基于 Windows 的模型化图形输入，使用户可以把更多的精力投入到系统模型的构建，而非语言的编程上。但是，要在 Simulink 上做出有个性的仿真，必须掌握一定的编程方法，尤其是一些自定义模块和 S 函数的设计与应用。

7.1.1　Simulink 启动

Simulink 的启动有两种方式：一种是启动 MATLAB 后，单击 MATLAB 主窗口的快捷按钮  ，打开 Simulink Library Brower 窗口，如图 7.1 所示。另一种方式是在 MATLAB 命令窗口中输入 simulink，然后按下回车键，弹出 Simulink Library Brower 窗口。

另外，在 MATLAB 命令窗口中输入 simulink3，结果是在桌面上出现一个用图标形式显示的 Library: simulink3 的 Simulink 模块库窗口，如图 7.2 所示。这两种

模块窗口界面只是不同的显示形式，用户可以根据自己喜好进行选择。一般来说，图 7.1 所示窗口直观、形象，易于初学者使用，但使用时会打开太多的子窗口。

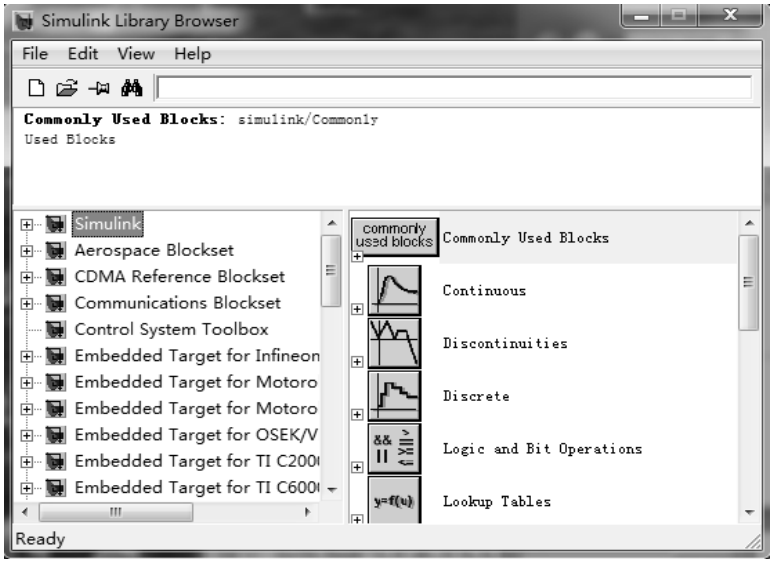


图 7.1　Simulink 模块库浏览界面

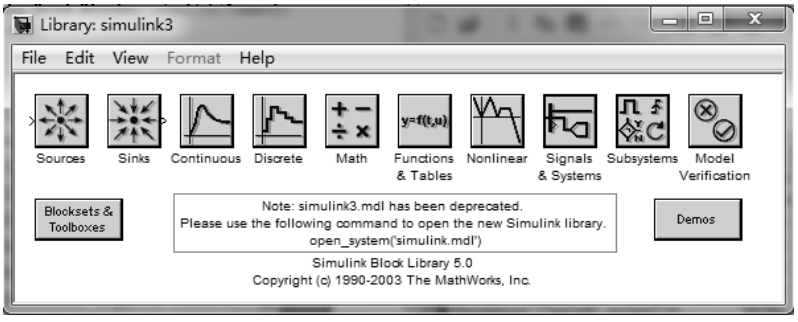


图 7.2　Simulink3 模块库浏览界面

Simulink 启动后，在 Simulink Library Browser 窗口菜单中，选择 File→New→Model，或者单击 File 菜单下第一个快捷键 🗋，便能够新建一个未命名的仿真编辑窗口，如图 7.3 所示。

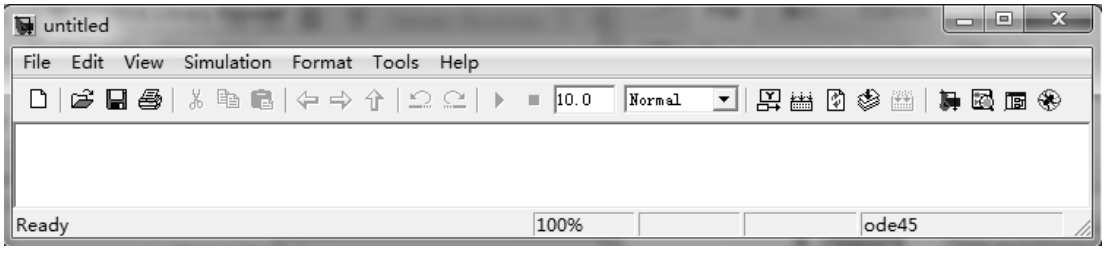


图 7.3　Simulink 仿真编辑窗口

7.1.2 Simulink 仿真设置

在图 7.3 中建立好模型，编辑好程序之后，需要设置仿真操作参数。单击 Simulation 菜单下面的“Configuration Parameters”选项或者直接按快捷键 Ctrl+E，便弹出如图 7.4 所示的设置界面，它包括仿真器参数（Solver）设置、工作空间数据导入/导出（Data Import/Export）设置、优化（Optimization）设置、诊断参数（Diagnostics）设置、硬件实现（Hardware Implementation）设置、模型引用（Model Referencing）设置等。

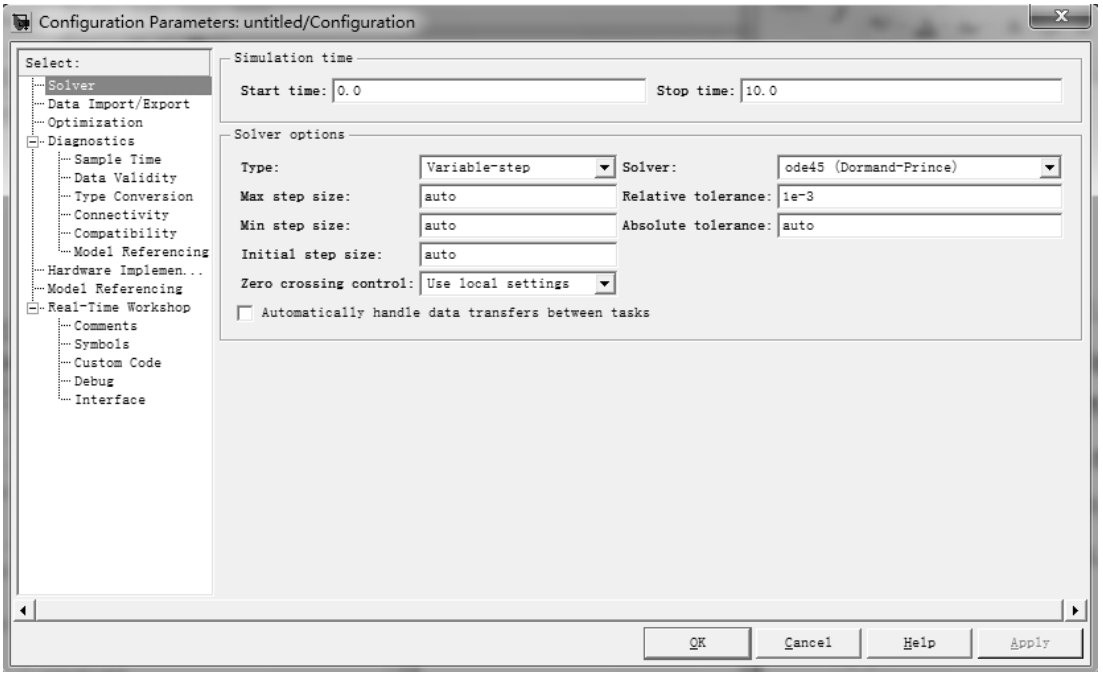


图 7.4　Simulink 设置界面

对于一般的仿真应用而言，这些设置都不需要改动，使用默认的设置便可以进行仿真。虽然设置的项很多，常用的则没有几个，下面分别介绍。

1．仿真器参数（Solver）设置

仿真参数设置窗口如图 7.5 所示，它可用于仿真开始时间、仿真结束时间、选择解法器及输出项等的选择。

（1）仿真时间

注意这里的时间概念与真实的时间并不一样，只是计算机仿真中对时间的一

种表示。比如，10s 的仿真时间，如果采样步长定为 0.1，则需要执行 100 步，若把步长减小，则采样点数增加，那么实际的执行时间就会增加。一般仿真开始时间设为 0，而结束时间视不同的因素而选择。总的说来，执行一次仿真要耗费的时间依赖于很多因素，包括模型的复杂程度、解法器及其步长的选择、计算机时钟的速度等。

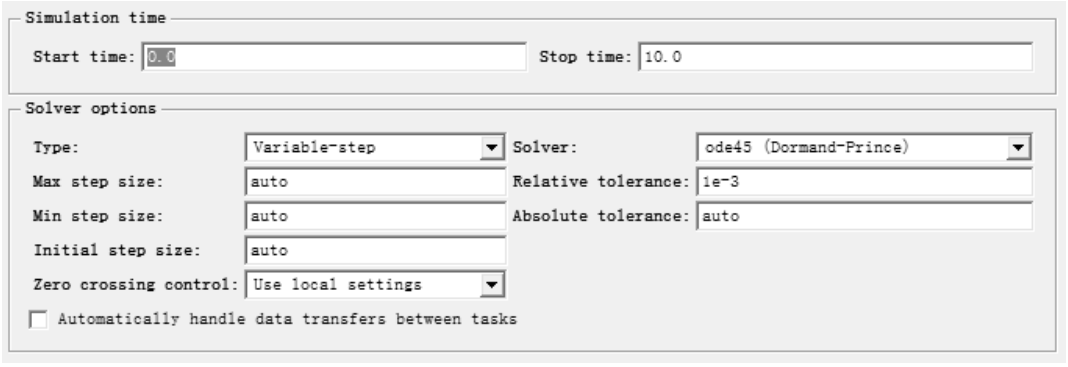


图 7.5　仿真参数设置窗口

（2）仿真步长模式

用户在 Type 后面的第一个下拉选项框中指定仿真的步长选取方式，可供选择的有 Variable-step（变步长）和 Fixed-step（固定步长）方式。变步长模式可以在仿真的过程中改变步长，提供误差控制和过零检测。固定步长模式在仿真过程中提供固定的步长，不提供误差控制和过零检测。用户还可以在第二个下拉选项框中选择对应模式下仿真所采用的算法。

变步长模式解法器有 ode45、ode23、ode113、ode15s、ode23s、ode23t、ode23tb 和 discrete。

ode45：默认值，四/五阶龙格-库塔法，适用于大多数连续或离散系统，但不适用于刚性（stiff）系统。它是单步解法器，也就是，在计算 $y(t_n)$ 时，它仅需要最近处理时刻的结果 $y(t_{n-1})$。一般来说，面对一个仿真问题最好是首先试试 ode45。

ode23：二/三阶龙格-库塔法，它在误差限要求不高和求解的问题不太难的情况下，可能会比 ode45 更有效。ode23 也是一个单步解法器。

ode113：是一种阶数可变的解法器，它在误差容许要求严格的情况下通常比 ode45 有效。ode113 是一种多步解法器，也就是在计算当前时刻输出时，它需要以前多个时刻的解。

ode15s：是一种基于数字微分公式的解法器（NDFs）。ode15s 也是一种多步解

法器。适用于刚性系统，当用户估计要解决的问题是比较困难的，或者不能使用 ode45，或者即使使用效果也不好，就可以用 ode15s。

ode23s：它是一种单步解法器，专门应用于刚性系统，在弱误差允许下的效果好于 ode15s。它能解决某些 ode15s 所不能有效解决的 stiff 问题。

ode23t：是梯形规则的一种自由插值实现。这种解法器适用于求解适度 stiff 的问题而用户又需要一个无数字振荡的解法器的情况。

ode23tb：是 TR-BDF2 的一种实现。TR-BDF2 是具有两个阶段的隐式龙格-库塔公式。

discrete：当 Simulink 检查到模型没有连续状态时使用它。

固定步长模式解法器有 ode5、ode4、ode3、ode2、ode1 和 discrete。

ode5：默认值，是 ode45 的固定步长版本，适用于大多数连续或离散系统，不适用于刚性系统。

ode4：四阶龙格-库塔法，具有一定的计算精度。

ode3：固定步长的二/三阶龙格-库塔法。

ode2：改进的欧拉法。

ode1：欧拉法。

discrete：是一个实现积分的固定步长解法器，它适合于离散无连续状态的系统。

（3）步长参数

对于变步长模式，用户可以设置最大的和推荐的初始步长参数。默认情况下，步长自动地确定，它由值 auto 表示。

Maximum step size（最大步长参数）：决定了解法器能够使用的最大时间步长，默认值为“仿真时间/50”，即整个仿真过程中至少取 50 个取样点。但对于仿真时间较长的系统，这样的取法则可能造成取样点过于稀疏，而使仿真结果失真。一般建议对于仿真时间不超过 15s 的系统，采用默认值即可；对于超过 15s 的每秒至少保证 5 个采样点；对于超过 100s 的系统，每秒至少保证 3 个采样点。

Initial step size（初始步长参数）：一般建议使用“auto”默认值即可。

（4）仿真精度的定义

Relative tolerance（相对误差）：指误差相对于状态的值，是一个百分比，默认值为 1e-3，表示状态的计算值要精确到 0.1%。

Absolute tolerance（绝对误差）：表示误差值的门限，或者是说在状态值为零的情况下，可以接受的误差。如果它被设成了 auto，那么 Simulink 为每一个状态设置初始绝对误差为 1e-6。

（5）Mode（固定步长模式选择）

Multitasking：选择这种模式时，当 Simulink 检测到模块间非法的采样速率转换，会给出错误提示。所谓非法采样速率转换指两个工作在不同采样速率的模块之间的直接连接。在实时多任务系统中，如果任务之间存在非法采样速率转换，那么就有可能出现一个模块的输出在另一个模块需要时却无法利用的情况。通过检查这种转换，Multitasking 将有助于用户建立一个符合现实的多任务系统的有效模型。使用速率转换模块可以减少模型中的非法速率转换。

Simulink 提供了两个这样的模块：unit delay 模块和 zero-order hold 模块。对于从慢速率到快速率的非法转换，可以在慢输出端口和快输入端口插入一个单位延时 unit delay 模块。而对于快速率到慢速率的转换，则可以插入一个零阶采样保持器 zero-order hold。

Singletasking：这种模式不检查模块间的速率转换，它在建立单任务系统模型时非常有用，在这种系统中就不存在任务同步问题。

Auto：在这种模式，Simulink 会根据模型中模块的采样速率是否一致，自动决定切换到 multitasking 和 singletasking。

（6）输出选项

Refine output：这个选项可以理解成精细输出，其意义是在仿真输出太稀松时，Simulink 会产生额外的精细输出，这一点就像插值处理一样。用户可以在 refine factor 设置仿真时间步间插入的输出点数。

要产生更光滑的输出曲线，改变精细因子比减小仿真步长更有效。精细输出只能在变步长模式中才能使用，并且在 ode45 效果最好。

Produce additional output：这个选项允许用户直接指定产生输出的时间点。一旦选择了该项，则在它的右边会出现一个 output times 编辑框，在这里用户指定额外的仿真输出点，它既可以是一个时间向量，也可以是表达式。与精细因子相比，这个选项会改变仿真的步长。

Produce specified output only：这个选项的意思是让 Simulink 只在指定的时间点上产生输出。为此解法器要调整仿真步长以使之和指定的时间点重合。这个选

项在比较不同的仿真时可以确保它们在相同的时间输出。

2．工作空间数据导入/导出设置

工作空间数据导入/导出设置窗口如图 7.6 所示，它主要用于在 Simulink 与 MATLAB 工作空间交换数值的有关选项设置，包括 Load from workspace、Save to workspace 和 Save options 三个选项。

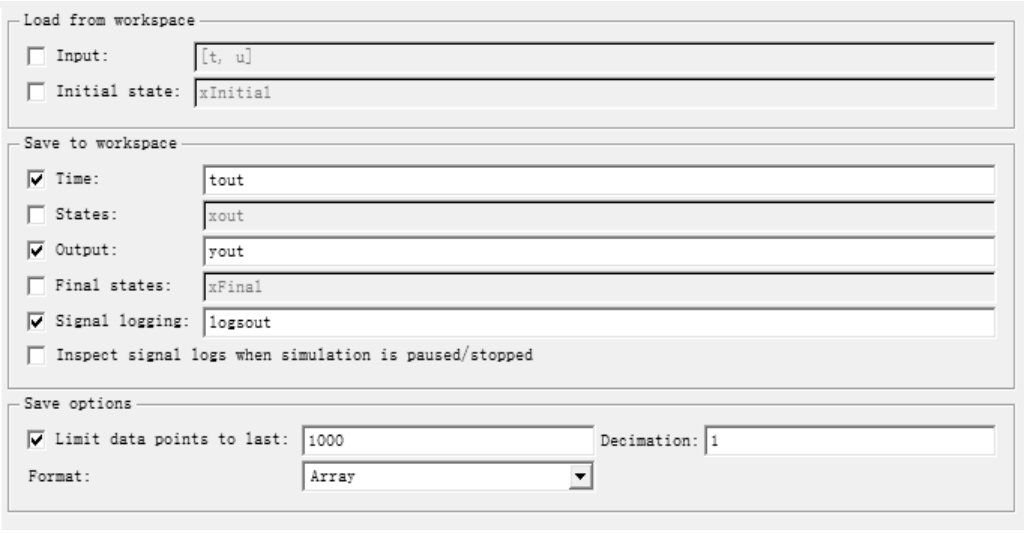


图 7.6　工作空间数据导入/导出设置窗口

Load from workspace：选中前面的复选框，即可从 MATLAB 工作空间获取时间和输入变量。一般，时间变量定义为 t，输入变量定义为 u。Initial state 用来定义从 MATLAB 工作空间获得的状态初始值的变量名。

Save to workspace：用来设置存在 MATLAB 工作空间的变量类型和变量名，选中变量类型前的复选框使相应的变量有效。一般存在工作空间的变量包括输出时间向量（Time）、状态向量（States）和输出变量（Output）。Final states 用来定义将系统稳态值存往工作空间时所用的变量名。

Save options：用来设置存往工作空间的有关选项。Limit data points to last 用来设置 Simulink 仿真结果最终可存往 MATLAB 工作空间的变量的规模，对于向量而言即其维数，对于矩阵而言即其秩。Decimation 设置了一个亚采样因子，它的默认值为 1，也就是对每一个仿真时间点产生值都保存，而若为 2，则是每隔一个仿真时刻才保存一个值。Format 用来说明返回数据的格式，包括矩阵 matrix、结构 struct 及带时间的结构 struct with time。

3．诊断参数设置

诊断参数设置主要包括采样时间、数据有效性、类型转换、连接性、兼容性、保存和模型引用这几个子项的诊断。用户可以设置当 Simulink 检查到这些子项事件时应做的处理，主要包括是否进行一致性检验、是否禁止复用缓存、是否进行不同版本的 Simulink 的检验等几项。

7.1.3　Simulink 模块库简介

标准的 Simulink 模块库中包括信号源模块组（Sources）、输出池模块组（Sinks）、连续模块组（Continuous）、离散模块组（Discrete）、非线性模块组（Discontinuities）、信号路径模块组（Signal Routing）、信号属性模块组（Signal Attributes）、数学运算模块组（Math Operations）、逻辑与位运算模块组（Logic and Bit Operations）、查表模块组（Lookup Tables）、用户自定义模块组（User-Defined Function）和端口与子系统模块组（Ports & Subsystems）等几个部分。这些模块组都是在系统建模时常用的模块。实际上，除此之外还有很多功能模块，用户也可以将自己编写的功能模块下挂到 Simulink 模块库中。本节将概括地对常用的几个模块进行介绍，在后面的建模过程中遇到相应的模块时会再作介绍。

下面简单介绍一下几组常用的模块组的主要模块功能。

1．信号源模块组（Sources）

信号模块组主要为系统提供各种信号源，常用的模块如图 7.7 所示。

（1）Sine Wave：生成正弦波，是比较常用的信号发生源。

（2）Ramp：生成斜坡信号。

（3）Step：生成阶跃信号。

（4）Chirp Signal：生成一个频率递增的正弦波。

（5）Random Number：生成高斯分布的随机信号，是比较常用的信号发生源。

（6）Uniform Random Number：生成均匀分布的随机信号。

（7）Constant：常值输入，产生一个常量信号，比较常用。

（8）From File：从外文件中读取数据。

（9）From Workspace：从 MATLAB 工作空间中读取数据。

其他模块如需要，可以单击对应模块右键查看 help for the XXX block，在 help

中查看其中的模块说明。如图 7.8 所示就是 Constant 的使用说明介绍。

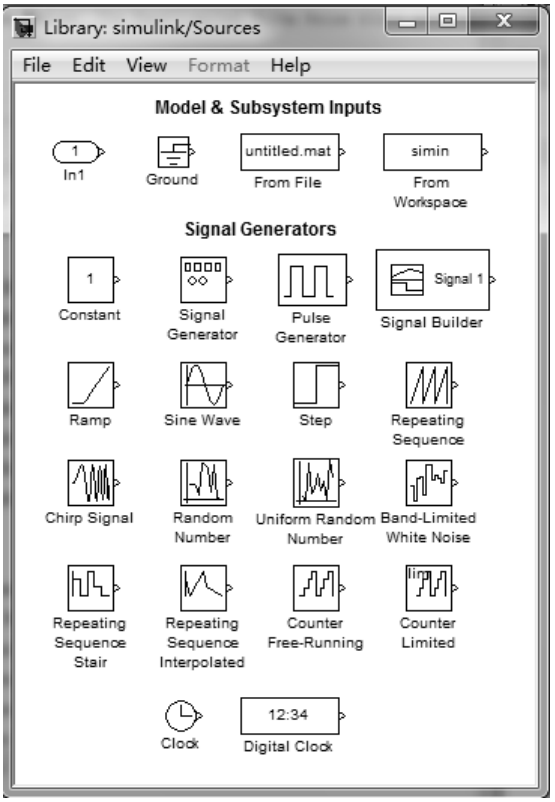


图 7.7　信号源模块

2．输出池模块组（Sinks）

输出池模块组为仿真提供输出元件的，用于接收、显示、输出仿真的结果。常用的输出模块与功能如下。

（1）Scope：显示仿真周期内产生的信号，相当于一个示波器。

（2）XY Graph：使用 MATLAB 图形窗口显示输出信号的 X-Y 图。

（3）Display：显示输出值。

（4）Out1：为模型或子系统提供一个输出池。

（5）Terminator：终止输出信号（为了防止一个输出没有连接到输出池时仿真出现警告的情况）。

（6）Stop Simulation：当系统输出为非零时停止仿真。

其他模块如需要，可以单击对应模块右键查看帮助说明。

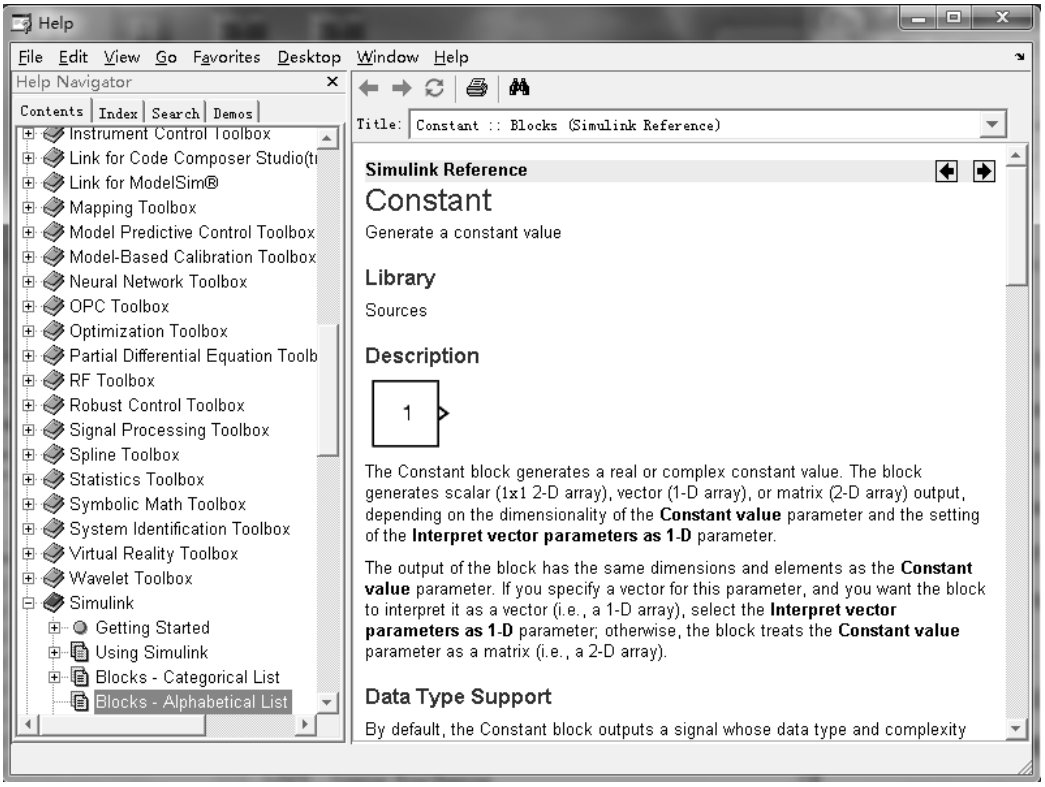


图 7.8　Constant 的使用说明

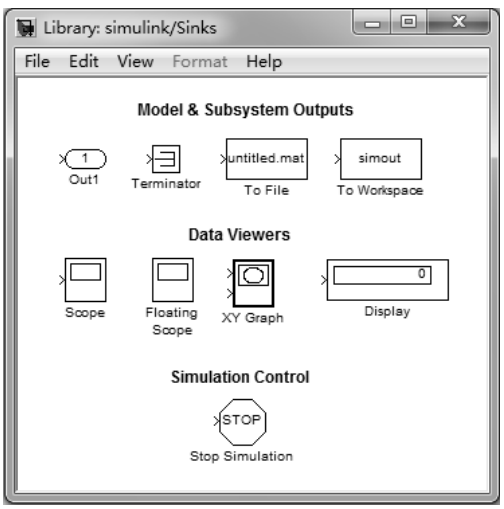


图 7.9　输出池模块组

3．数学运算模块组（**Math Operations**）

数学运算模块组为仿真提供数学运算功能模块，最常用的如信号做加减运算、

求和运算等。常用的数学运算模块如下。

（1）Sum：求和运算。

（2）Add：加法。

（3）Bias：偏移。

（4）Abs：取输入值的绝对值。

（5）Sqrt：输入值开根号。

（6）Product：计算输入值的简积或商。

（7）Dot Product：进行点积运算。

（8）Sign：用一个符号函数对输入值的正负性作出判断。

（9）Rounding Function：执行圆整函数。

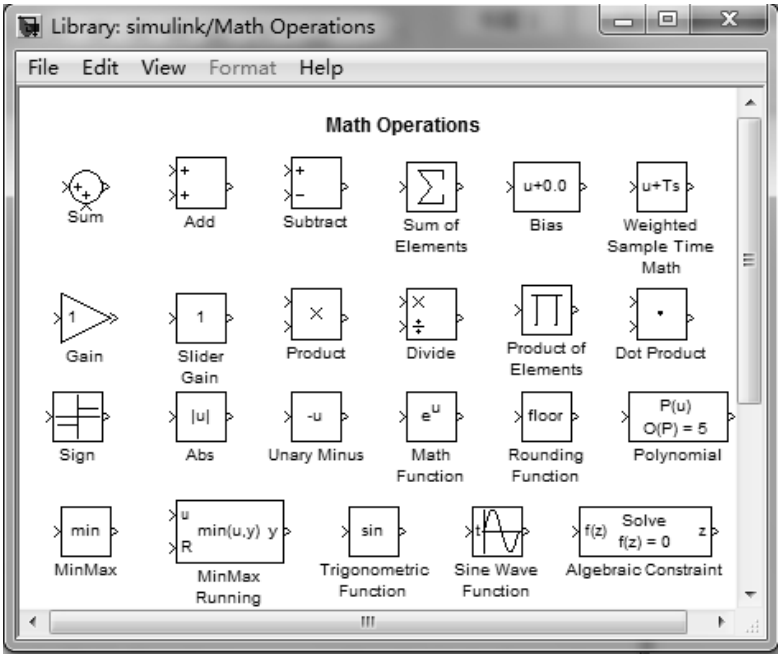


图 7.10　数学运算模块组

数学运算模块的其他模块，可以单击对应模块右键查看帮助说明。

4．用户自定义函数模块组（User-Defined Functions）

再全再丰富的模块库，也无法满足用户各种需求，因此用户自定义模块显得非常必要。用户自定义函数模块组为仿真提供各种特色模块，能设定比较复杂的功能模块，还能利用用户自己编写的功能函数。

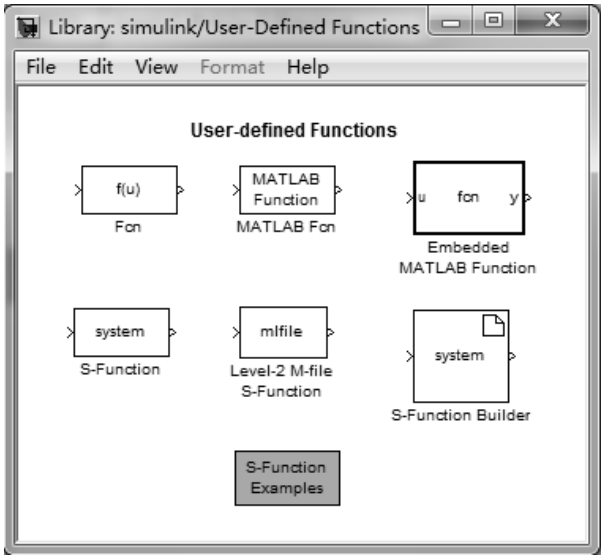


图 7.11　用户自定义函数模块组

（1）Fcn：用自定义的函数（表达式）进行运算。

（2）InterpretedMATLAB Function ：利用 MATLAB 的现有函数进行运算。

（3）MATLAB Function：嵌入的 MATLAB 函数。

（4）S-Function: 调用自编的 S 函数程序进行运算，在后文重点介绍。

（5）Level-2 M-file S-Function：M 文件编写的 S 函数。

（6）S-Function Builder：S 函数建立器。

上面介绍的是 Simulink 模块库中常用的模块。除此之外，Simulink 还提供了许多许多功能强大的模块集，比如航天模块集、通信系统仿真模块集、数字信号处理模块集，等等。这些模块集在解决实际仿真问题时起到了很大的作用，用户可以根据自己的需求选择，在这就不一一介绍。

7.2　S 函数

S 函数（S-Function）是系统函数（System Function）的简称，是一个动态系统的计算机语言描述。在 MATLAB 中，用户可以选择用 M 文件编写，也可以选择用 C 语言或者 HEX 文件编写。

7.2.1　S 函数原理

S 函数之所以会出现，是因为在研究中经常需要复杂的算法设计，而这些算法

因为其复杂性不适合用普通的 Simulink 模块来搭建，即 MATLAB 所提供的 Simulink 模块不能满足用户的需求，需要用编程的形式设计出 S 函数模块，然后将其嵌入到系统中。如果恰当使用 S 函数，理论上讲可以在 Simulink 下对任意复杂的系统进行仿真。本书主要介绍目标定位跟踪算法，涉及各种滤波算法，那么必然需要使用 S 函数，将算法过程固化在 S 函数模块中，这样在 Simulink 下仿真就容易调用了。

S 函数有固定的程序格式，用 MATLAB 语言可以编写 S 函数，此外还允许用户使用 C、C++、Fortran 和 Ada 等语言进行编写。用非 MATLAB 语言进行编写时，需要采用编译器生成动态链接库 DLL 文件。在命令窗口中输入 sfundemos，或者单击 Simulink→User-Defined Functions→S-Function Examples，即可出现如图 7.12 所示的界面，可以选择对应的编程语言查看演示文件。

MATLAB 为了用户使用方便，有一个 S 函数的模板 sfuntmpl.m。一般来说，我们仅需要在 sfuntmpl.m 的基础上进行修改即可。在命令窗口输入 edit sfuntmpl 即可出现模板函数的内容，可以详细地观察其帮助说明以便更好地了解 S 函数的工作原理。

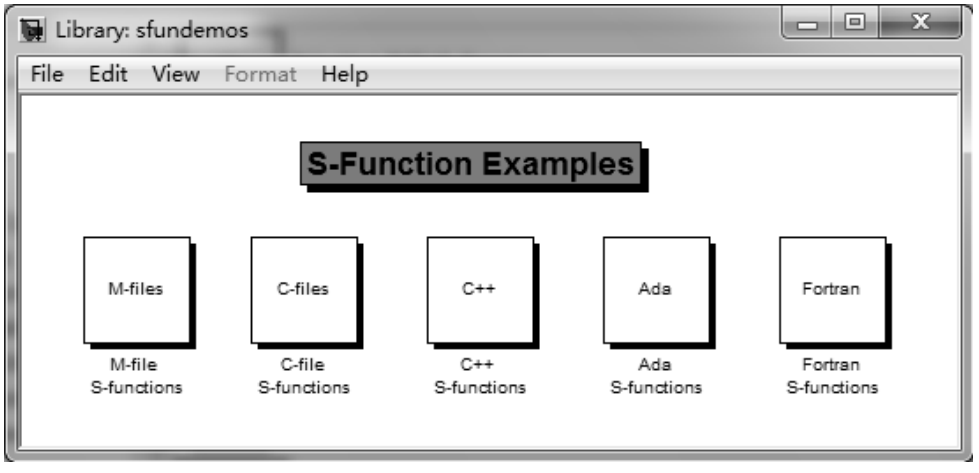


图 7.12　S 函数范例

模板函数的定义形式为

```
function [sys,x0,str,ts]=sfuntmpl(t,x,u,flag)
```

去除全部英文注释，同时将该函数加入中文注解如下：

```
%%%
```

```
sizes.NumInputs      = 0;   % 输入数目
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;    % 至少需要的采样时间
sys = simsizes(sizes);
x0  = [];                    % 初始条件
str = [];                    % str 总是设置为空
ts  = [0 0];                 % 初始化采样时间
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2．进行连续状态变量的更新
function sys=mdlDerivatives(t,x,u)
sys = [];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3．进行离散状态变量的更新
function sys=mdlUpdate(t,x,u)
sys = [];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4．求取系统的输出信号
function sys=mdlOutputs(t,x,u)
sys = [];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 5．计算下一仿真时刻，由 sys 返回
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;      %   此处设置下一仿真时刻为 1 秒钟以后
sys = t + sampleTime;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 6．结束仿真子函数
function sys=mdlTerminate(t,x,u)
sys = [];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

一般来说，S 函数的定义形式为

```
function [sys,x0,str,ts]=sfunc(t,x,u,flag,P1,…Pn)
```

其中的 sfunc 为自己定义的函数名称。输入参数 t、x、u，分别对应时间、状态、输入信号；flag 为标志位，其取值不同，S 函数执行的任务和返回的数据也是不同的；Pn 为额外的参数。输出参数 sys 为一个通用的返回参数值，其数值根据 flag 的不同而不同；x0 为状态初始数值；str 在目前为止的 MATLAB 版本中并没有什么作用，因此总是将 str 置空；ts 为一个两列的矩阵，包含采样时间和偏移量两个参数，如果设置为[0 0]，那么每个连续的采样时间步都运行，[-1 0]则表示按照所连接的模块的采样速率进行，[0.25 0.1]表示仿真开始的 0.1s 后每 0.25s 运行一次，采样时间点为 TimeHit=n*period+offset。

S 函数的使用过程中有两个概念值得注意，具体如下。

（1）DirFeedthrough，见初始化子函数，系统的输出是否直接和输入相关联，即输入是否出现在输出端的标志，若是则为 1，否则为 0。一般可以根据在 flag=3 的时候，mdlOutputs 函数是否调用输入 u 来判断是否直接馈通。

（2）dynamically sized inputs，见初始化子函数，主要给出连续状态的个数、离散状态的个数、输入数目、输出数目等。

S 函数中目前支持的 flag 选择有 0、1、2、3、4、9 等几个数值。下面说一下在不同的 flag 情况下 S 函数的执行情况。

（1）flag=0。进行系统的初始化过程，调用 mdlInitializeSizes 函数，对参数进行初始化设置，如离散状态个数、连续状态个数、模块输入和输出的路数、模块的采样周期个数、状态变量初始数值等。

（2）flag=1。进行连续状态变量的更新，调用 mdlDerivatives 函数。

（3）flag=2。进行离散状态变量的更新，调用 mdlUpdate 函数。

（4）flag=3。求取系统的输出信号，调用 mdlOutputs 函数。

（5）flag=4。调用 mdlGetTimeOfNextVarHit 函数，计算下一仿真时刻，由 sys 返回。

（6）flag=9。终止仿真过程，调用 mdlTerminate 函数。

7.2.2　S 函数的控制流程

S 函数的调用顺序是通过 flag 标志来控制的。在仿真初始化阶段，通过设置 flag 标志为 0 来调用 S 函数，并请求提供数量，主要包括连续状态、离散状态、输入和输出的个数、初始状态、采样时间等。接下来 flag 标志设为 3，请求 S 函数计

算模块的输出。然后设置 flag 标志为 2，更新离散状态。当用户还需要计算状态导数时，可设置 flag 标志为 1，由求解器使用积分算法计算状态的值。计算出状态导数和更新离散状态之后，通过设置 flag 标志为 3 来计算模块的输出，这样就结束了一个仿真周期。最后通过不断循环上述过程，到达仿真结束时间，这时设置 flag 标志为 9，结束仿真。这个过程如图 7.13 所示。

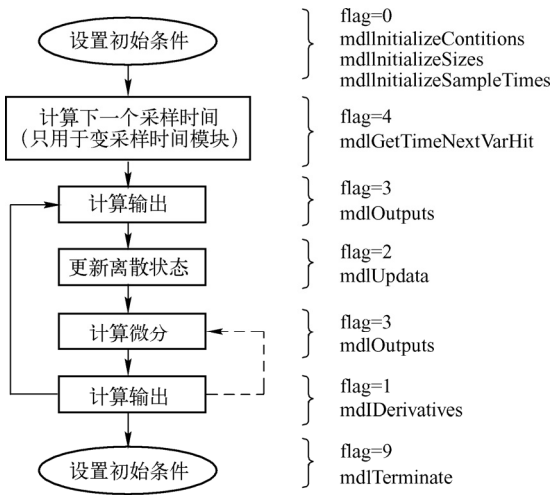


图 7.13　S 函数控制流程

在 S 函数的编写过程中，首先需要搞清楚模块中有多少个连续和离散状态，离散模块的采样周期是多少，同时需要了解模块的连续和离散的状态方程分别是什么，输出如何表示。

7.3　线性 Kalman 的 Simulink 仿真

7.3.1　一维数据的 Kalman 滤波处理

温度、距离、信号强度、物体质量等的测量都可以看作一维数据的测量。在工程实际中，一维数据测量是最常见的，对一维数据的滤波平滑也是最常用的数据加工处理方法。本节选取雷达对远山的一维距离数据的测量为例，来说明 Simulink 下的 Kalman 滤波问题。

假设雷达要测量其与远处一座山峰的距离，雷达的位置可以认为是（0,0），而远山处于雷达的正前方，位置为（5000,0），那么两者之间的距离大约为 5000m。

建立坐标系如图 7.14 所示。

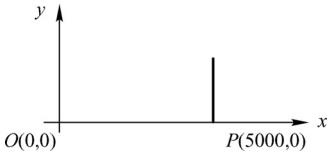


图 7.14　雷达与远山坐标示意图

由于远山是静止的，故状态方程可以写为

$$x(k+1)=Fx(k)+w(k)$$

这里的 F=1，而过程噪声为 0，可以省略 $w(k)$。而状态 x 表示远山的 x 坐标轴位置。而雷达对目标测量，其方程为

$$z(k)=Hx(k)+v(k)$$

这里测量值是距离，等同于目标的 x 轴位置，故 H=1，雷达的测量噪声 $v(k)$ 的均值为 0，方差为 5。那么对该系统用 Simulink 仿真过程如下。

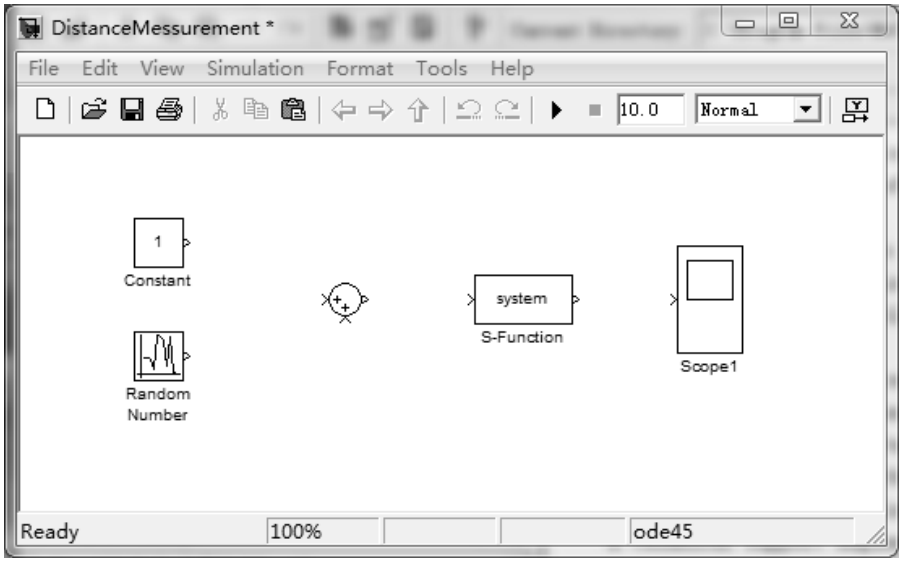


图 7.15　仿真模块

第一步：建模。启动 MATLAB，单击 Simulink，在弹出的 Simulink Library Browser 窗口中选择 File→New→Model，这样就新建了一个仿真编辑窗口，同时按 Ctrl+S 键把它保存为 DistanceMessurement，在 Sources 库中找到 Constant 模块和 Random Number 模块，在 Sinks 库中找到 Scope 模块，在 Math Operations 库中

找到求和模块，在 User-Define Functions 库中找到 S-Function 模块，将它们都拖入 DistanceMessurement 窗口中，如图 7.15 所示。

第二步：修改参数。将 Constant 模块双击或者右键选择 Constant Parameters，在弹出的 Source Block Parameters:Constant 窗口中修改 Constant value 值设置为 5000；同样的操作将 Random Number 模块的均值设为 0，方差设为 5。双击 Scope1 模块，在弹出的窗口中单击菜单快捷工具 Parameters，设置坐标轴 Number of axes 的数值为 3。最后将所有的模块按照图 7.16 链接起来。

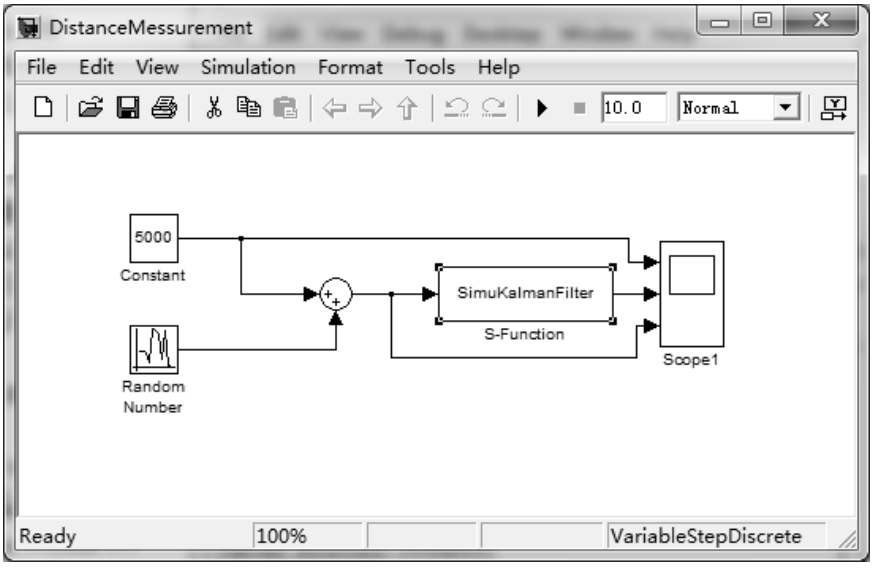


图 7.16(a)　系统建模

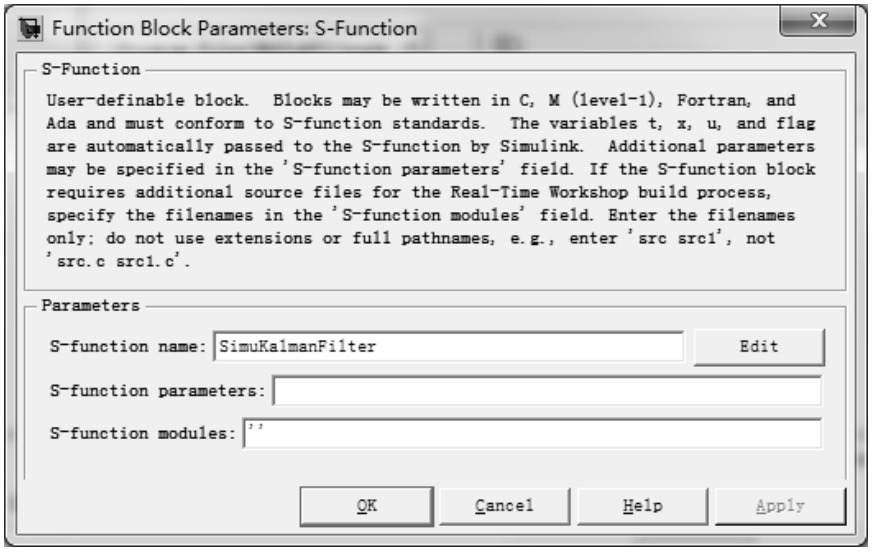


图 7.16(b)　S 函数名称修改

第三步：双击 S-Function 模块，在弹出的窗口中的如图 7.16(b)所示，在 S-Function name 文本框中输入 SimuKalmanFilter（在这之前，请在 MATLAB 的工作目录 C:\Progam Files\MATLAB71\work 下创建一个 SimuKalmanFilter.m 文件），然后单击 Edit 按钮，这时候我们需要对 S 函数编辑，实现 Kalman 对输入信号滤波。新建一个 M 文件，在 SimuKalmanFilter.m 文件中输入以下代码。

```
%%%
```

```
sizes.NumDiscStates   = 1;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;    % 至少需要的采样时间
sys = simsizes(sizes);
x0   = 5000+sqrt(5)*randn;                % 初始条件
str = [];                          % str 总是设置为空
ts   = [-1 0];   % 表示该模块采样时间继承其前的模块采样时间设置
global P;    % 定义协方差 P
P=5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2．进行连续状态变量的更新
function sys=mdlDerivatives(t,x,u)
sys = [];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3．进行离散状态变量的更新
function sys=mdlUpdate(t,x,u)
global P;
F=1;   % 系统的状态方程和观测方程系数矩阵
B=0;
H=1;   % 系统的状态方程和观测方程系数矩阵
Q=0;   % 过程噪声和测量噪声方差
R=5;
xpre=F*x+B*u; % 状态预测
Ppre=F*P*F'+Q;% 协方差预测
K=Ppre*H'*inv(H*Ppre*H'+R);% 计算 Kalman 增益
e=u-H*xpre;    % u 是输入的观测值，在此计算新息
xnew=xpre+K*e; % 状态更新
P=(eye(1)-K*H)*Ppre; % 协方差更新
% 将计算的结果返回给主函数
sys=xnew;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4．求取系统的输出信号
function sys=mdlOutputs(t,x,u)
sys = x;   % 把算得的模块输出向量赋给 sys
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 5．计算下一仿真时刻，由 sys 返回
function sys=mdlGetTimeOfNextVarHit(t,x,u)
```

```
sampleTime = 1;       %   此处设置下一仿真时刻为 1s 以后
sys = t + sampleTime;
%%%
```

注意：Scope 模块在仿真结束时，曲线有时无法正常显示在面板中间，这时可以单击 Scope 工具栏的 Autoscale 快捷工具，这时曲线会移到合适的视图。为了对比曲线效果，将 *Y* 轴范围统一，在 Scope 中图形显示区域单击右键→Axes properties，在弹出的对话框中输入 *Y* 轴的最大最小值，如图 7.19 所示，将其他两个曲线坐标轴执行同样的操作，则能得到图 7.18 所示的效果。

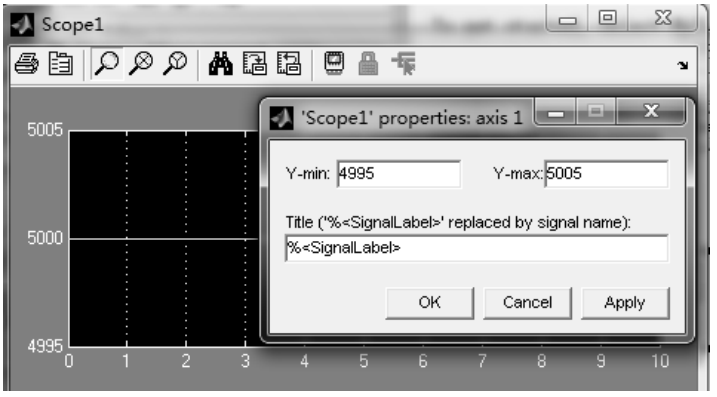


图 7.19　统一坐标 *Y* 的参数修改

从图 7.18 中可以看出，雷达与远山的真实距离为 5000m，但是实际测量值因为受到仪器的测量误差的影响，表现了很大的波动。经过 Kalman 滤波后，数据又变得平滑且最终稳定收敛在 5000m。可见 Kalman 滤波是有效的，它大大降低了测量噪声的扰动。

7.3.2　状态方程和观测方程的 Simulink 建模

状态空间模型是控制系统最新与最科学的描述方法。在电路分析、电机控制、目标跟踪等一切与时间序列相关的模型中，在 MATLAB 下都可以通过构建系统的状态方程和观测方程并进行仿真。

在 Simulink 下，也可以通过 Continuous 库或者 Discrete 库中 State-Space 模块来构建线性系统的状态方程和观测方程，如图 7.20 所示，通过设置 A、B、C、D 四个矩阵，便可以方便地表达任何线性系统的模型。但是，很多非线性系统并不能提出确切提出系数矩阵 A、B、C、D，那么这时笔者建议用 S 函数来完成状态方程和观测方程的建模。因为 S 函数是自定义的，可以根据需要，表示任何形式的状态空间。

下面，以目标跟踪为例，介绍如何构建系统的状态方程和观测方程的 Simulink 模型。

假定一个目标做匀速直线运动，状态方程为

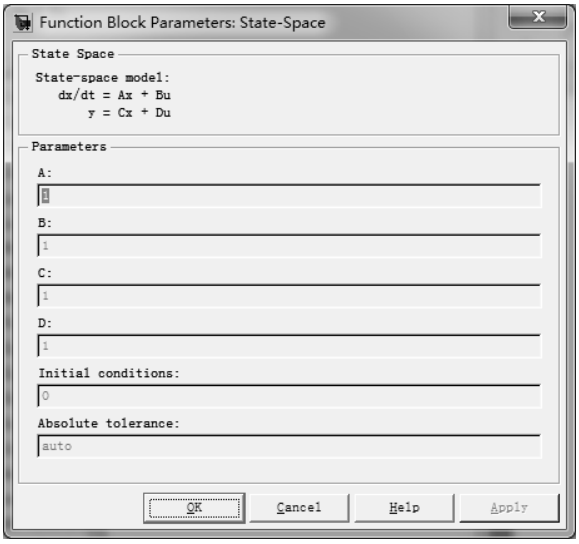


图 7.20　State-Space 模块的参数

$$\boldsymbol{X}(k+1)=\boldsymbol{F}*\boldsymbol{X}(k)+\boldsymbol{G}*\boldsymbol{w}(k) \tag{7.1}$$

其中目标的状态为 $\boldsymbol{X}(\mathrm{k})=[x(k),\dot{x}(k),y(k),\dot{y}(k)]^{\mathrm{T}}$，初始时刻的值为 $\boldsymbol{X}(0)=[10,5,12,5]^{\mathrm{T}}$，目标的过程噪声 $\mathrm{w}(k)$ 的方差为 $\boldsymbol{Q}=\mathrm{diag}([0.01,0.09])$，即水平和垂直方向的速度噪声方差分别为 0.01 和 0.09，假如采样时间为 1s，则 F 和 G 分别为

$$\boldsymbol{F}=\begin{bmatrix}1 & \Delta t & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & \Delta t\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\end{bmatrix},$$

$$\boldsymbol{G}=\begin{bmatrix}0.5\Delta t^2 & 0\\ \Delta t & 0\\ 0 & 0.5\Delta t^2\\ 0 & \Delta t\end{bmatrix}=\begin{bmatrix}0.5 & 0\\ 1 & 0\\ 0 & 0.5\\ 0 & 1\end{bmatrix}$$

按照图 7.21，分别在 Simulink 仿真编辑窗口中拖入 Sources 库中的 Random Number 模块两个，Signal Routing 库中的 Mux 和 Demux 模块各 1 个，Sinks 库中

的 Floating Scope 模块 3 个和 XY Graph 模块 1 个，User-Defined Functions 库中的 S-Function 模块 1 个，将它们连接起来。

设置两个 Randon Number 模块的均值为 0，方差分别为 0.01 和 0.09，Initial seed 的值设为不一样即可。本例中将它们分别设置为 0 和 3，Sample time 都设置为 1 双击 S 函数模块，将 S-Function name 改成 SimuStateFunction，接下来编写 M 文件 SimuStateFunction.m。打开 M 文件编辑器，输入以下代码。

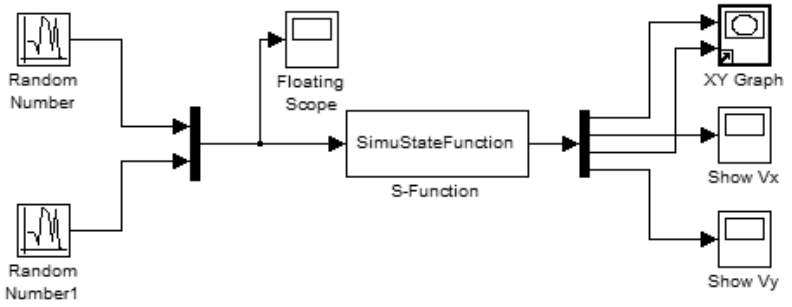


图 7.21　S 系统状态方程模型

```
%%%
```

```
sizes = simsizes;
sizes.NumContStates  = 0;    % 无连续量
sizes.NumDiscStates  = 4;    % 离散状态 4 维
sizes.NumOutputs     = 4;    % 输出 4 维，应为状态量是 x-y 方向的位置和速度
sizes.NumInputs      = 2;    % 输入维数，因为噪声模型是 2 维的
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;    % 至少需要的采样时间
sys = simsizes(sizes);
x0  = [10,5,12,5]';          % 初始状态
str = [];                    % str 总是设置为空
ts  = [-1 0];  % 表示该模块采样时间继承其前的模块采样时间设置
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2．进行离散状态变量的更新
function sys=mdlUpdate(t,x,u)
G=[0.5,0;1,0;0,0.5;0,1];  % 过程噪声驱动矩阵
F=[1,1,0,0;0,1,0,0;0,0,1,1;0,0,0,1]; % 状态转移矩阵
sys =F*x+G*u;  % 状态更新
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3．求取系统的输出信号
function sys=mdlOutputs(t,x,u)
sys = x;  % 把算得的模块输出向量赋给 sys
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

保存程序文件，将 Simulation stop time 改成 100，运行仿真模型，得到目标过程噪声如图 7.22 所示，运行轨迹图如 7.23 所示，*X* 方向和 *Y* 方向的速度分别如图 7.24 和图 7.25 所示。

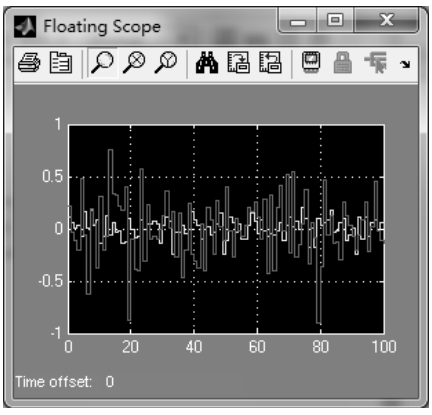


图 7.22　过程噪声模型

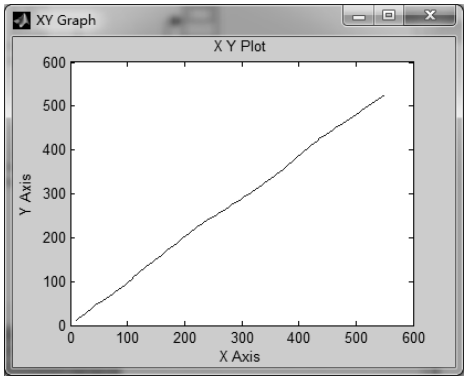


图 7.23　目标运行轨迹

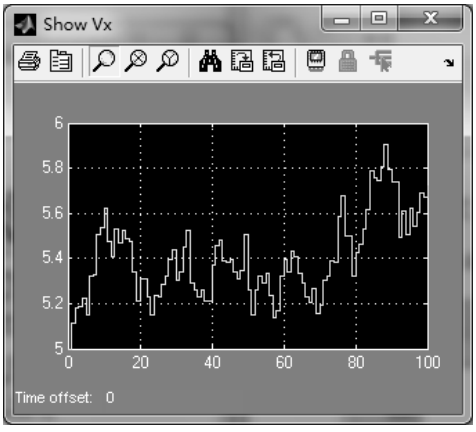


图 7.24　X 方向的速度

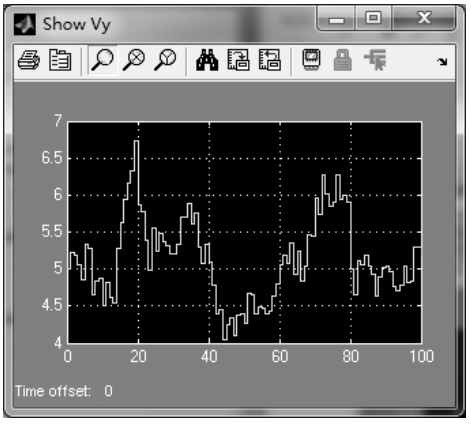


图 7.25　Y 方向的速度

接下来构建观测方程的模型，假定雷达对目标的位置测量，观测方程为

$$\boldsymbol{Z}(k)=\boldsymbol{H}*\boldsymbol{X}(k)+\boldsymbol{I}*\boldsymbol{v}(k) \tag{7.2}$$

式中，$\boldsymbol{H}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}$，$\boldsymbol{I}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$，而测量噪声 $v(k)$ 的的方差为 $\boldsymbol{R}=\mathrm{diag}([0.04,0.04])$。

在仿真编辑窗口中继续拖入 Random Number 模块 2 个，从 Math Operations 库中拖入 2 个 Sum 模块，2 个 Floating Scope 模块，如图 7.26 所示。同理将 Random Number2 和 Random Number3 模块的均值都设为 0，方差都设为 0.04，Initial seed 设置为不同的值即可，采样时间都设为 1。

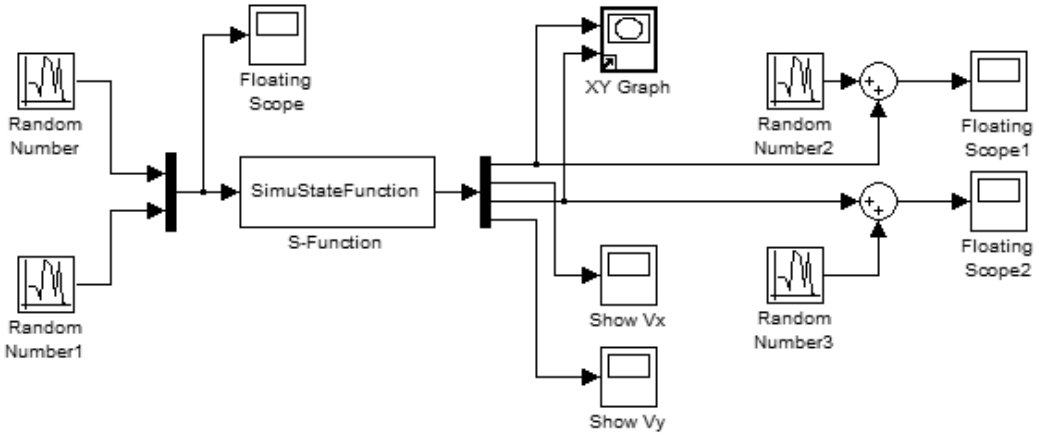


图 7.26　一个完整的系统方程建模

运行仿真，可以得到 X 和 Y 方向上位置，如图 7.27 和图 7.28 所示。

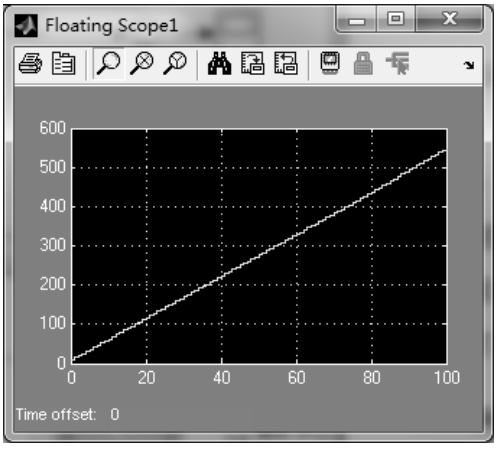


图 7.27　X 方向的位置

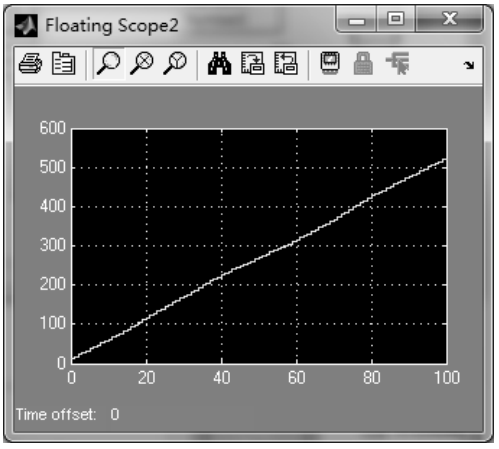


图 7.28　*Y* 方向的位置

至此，一个完整系统模型在 Simulink 环境下的建模已经完成。在这里的观测方程是线性的，如果要做非线性的观测方程，同样可以利用 S 函数完成建模。读者可以自己尝试。

7.3.3　基于 S 函数的 Kalman 滤波器设计

假定系统的数学模型如式（7.1）和式（7.2）及图 7.29 所示，构建目标跟踪系统。这里需要 4 个 Random Number 模块，两个 Mux 模块和两个 Demux 模块，两个求和 Sum 模块，7 个 Scope 模块，两个 XY Graph 模块，两个 S-Function 模块，将它们的链接起来。其中，Random Number 和 Random Number1 的方差分别为 0.0001 和 0.0009，均值都为 0，采样时间为 1；Random Number2 和 Random Number3 的方差都设置为 0.04，均值都为 0，采样时间都为 1。

最后重点是构建两个自定义模块，双击 S-Function 和 S-Function1 分别将它们 S-Function name 设置为 SimuStateFunction 和 KalmanFilter。

在 MATLAB 中编辑 M 文件 SimuStateFunction.m 和 KalmanFilter.m，分别如下所示。

（1）SimuStateFunction.m 文件

```
% 功能说明：S 函数仿真系统的状态方程 X(k+1)=F*x(k)+G*w(k)
function [sys,x0,str,ts]=SimuStateFunction(t,x,u,flag)
global Xstate;
switch flag
```

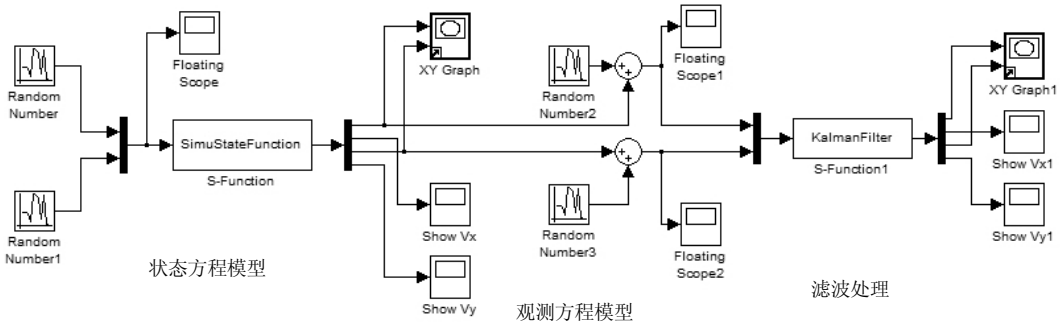


图 7.29　基于 Kalman 滤波的目标跟踪的整体模型

```
        case 0   % 系统进行初始化，调用 mdlInitializeSizes 函数
            [sys,x0,str,ts]=mdlInitializeSizes;
        case 2   % 更新离散状态变量，调用 mdlUpdate 函数
            sys=mdlUpdate(t,x,u);
        case 3   % 计算 S 函数的输出，调用 mdlOutputs
            sys=mdlOutputs(t,x,u);
        case {1,4}
            sys=[];
        case 9    % 仿真结束，保存状态值
            save('Xstate','Xstate');
        otherwise    % 其他未知情况处理，用户可以自定义
            error(['Unhandled flag = ',num2str(flag)]);
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1．系统初始化子函数
    function [sys,x0,str,ts]=mdlInitializeSizes
    sizes = simsizes;
    sizes.NumContStates   = 0;    % 无连续量
    sizes.NumDiscStates   = 4;    % 离散状态 4 维
    sizes.NumOutputs      = 4;    % 输出 4 维
    sizes.NumInputs       = 2;    % 输入维数，因为噪声模型是 2 维的
    sizes.DirFeedthrough = 0;
    sizes.NumSampleTimes = 1;    % 至少需要的采样时间
    sys = simsizes(sizes);
    x0   = [10,5,12,5]';                % 初始条件
    str = [];                    % str 总是设置为空
    ts   = [-1 0];   % 表示该模块采样时间继承其前的模块采样时间设置
```

```
global Xstate;
Xstate=[];
Xstate=[Xstate,x0];
%%%
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1．系统初始化子函数
function [sys,x0,str,ts]=mdlInitializeSizes
global P;
global Xkf;
global Z;     %
P=0.1*eye(4);
sizes = simsizes;
sizes.NumContStates  = 0;     % 无连续量
sizes.NumDiscStates  = 4;     % 离散状态 4 维
sizes.NumOutputs     = 4;     % 输出 4 维，应为状态量是 x-y 方向的位置和速度
sizes.NumInputs      = 2;     % 输入维数，因为噪声模型是 2 维的
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;     % 至少需要的采样时间
sys = simsizes(sizes);
x0   = [10,5,12,5]';                  % 初始条件
str = [];                             % str 总是设置为空
ts   = [-1 0];   % 表示该模块采样时间继承其前的模块采样时间设置
Xkf=[];
Z=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2．进行离散状态变量的更新
function sys=mdlUpdate(t,x,u)
global P;
global Xkf;
global Z;
F=[1,1,0,0;0,1,0,0;0,0,1,1;0,0,0,1]; % 状态转移矩阵
G=[0.5,0;1,0;0,0.5;0,1];                  % 过程噪声驱动矩阵
H=[1,0,0,0;0,0,1,0];                      % 观测矩阵
Q=diag([0.0001,0.0009]);                        % 过程噪声方差
R=diag([0.04,0.04]);                      % 测量噪声方差
% Kalman 滤波
Xpre=F*x;     % 状态预测
Ppre=F*P*F'+G*Q*G';% 协方差更新
K=Ppre*H'*inv(H*Ppre*H'+R); % 计算 Kalman 增益
e=u-H*Xpre; % 计算新息
Xnew=Xpre+K*e; % 状态更新
P=(eye(4)-K*H)*Ppre; % 协方差更新
```

```
sys=Xnew;     % 将计算的结果返回给主函数
% 保存观测值和滤波结果
Z=[Z,u];
Xkf=[Xkf,Xnew];
%%%
```

笔者认为，Simulink 的 Sinks 库中的显示模块，没有 MATLAB 中的 plot 函数使用灵活，Scope 和 XY Graph 做曲线展示的时候具有局限性。因此在做误差分析时候，本例中将目标运动的真实状态保存在 Xstate.mat 中，观测数据保存在 Zobserv.mat 中，Kalman 滤波的状态结果保存在 Xkalman.mat 中，这时候再通过编写误差分析函数 DeviationAnalysis.m 文件，对仿真数据做进一步的分析和处理。

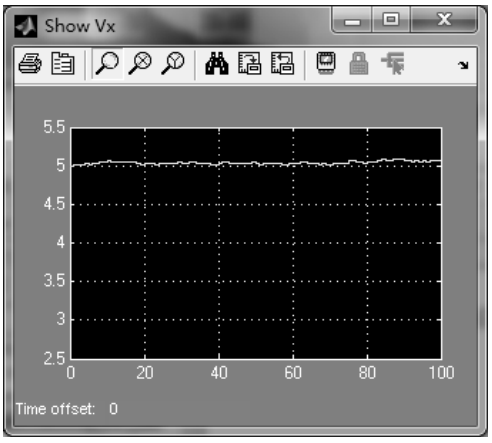


图 7.32　*X* 方向真实速度

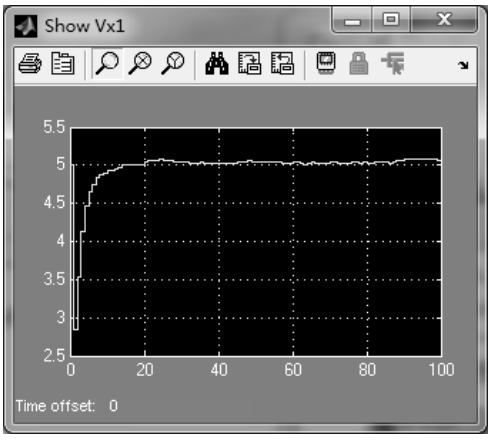


图 7.33　Kalman 滤波估计的 *X* 方向速度

```
%%%
```

```
% 计算误差
T1=length(Xstate(1,:))   % 虽然仿真时间设 100，但是 Xstate 的列数却为 102
T2=length(Z(1,:))        % 虽然仿真时间设 100，但是 Z 的列数却为 101
T=min(T1,T2)
Div_Observ_Real=zeros(1,T);
Div_Kalman_Real=zeros(1,T);
for i=2:T
    Div_Observ_Real(i)=sqrt( (Z(1,i)-Xstate(1,i))^2+(Z(2,i)-Xstate(3,i))^2 );
    Div_Kalman_Real(i)=sqrt( (Xkf(1,i)-Xstate(1,i))^2+(Xkf(3,i)-Xstate(3,i))^2 );
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 画轨迹图
figure
hold on;box on;
plot(Xstate(1,:),Xstate(3,:),'-r.');
plot(Xkf(1,:),Xkf(3,:),'-k+');
plot(Z(1,:),Z(2,:),'-k*');
% 偏差图
figure
hold on;box on;
plot(Div_Observ_Real,'-ko','MarkerFace','g');
plot(Div_Kalman_Real,'-ks','MarkerFace','b');
legend('Observ','Kalman');
```

运行数据分析程序，得到目标的轨迹图 7.34 和位置偏差图 7.35。从程序结果图可以看出，观测值和 Kalman 滤波结果值都比较好地逼近了目标运动的真实位置值，但是从图 7.35 中可以看出，与直接观测的位置偏差相比，Kalman 滤波后的位置偏差较小。

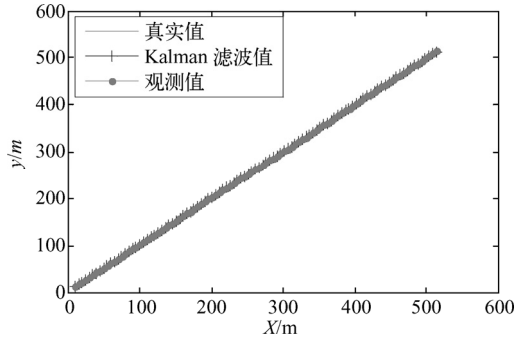


图 7.34　轨迹对比图

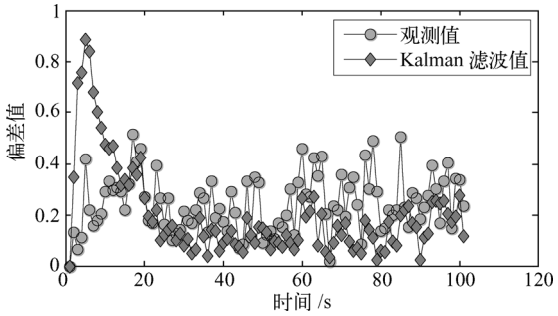


图 7.35　位置偏差图

注意本例中共有 7 个文件，分别是 DeviationAnalysis.m、KalmanFilter.m、SimuStateFunction.m、System_TargetTracking_KF_Simulation.mdl、Xkalman.mat、Xstate.mat、Zobserv.mat。其中，KalmanFilter.m 和 SimuStateFunction.m 是不能独立运行的，而.mat 文件是运行 System_TargetTracking_KF_Simulation.mdl 模块产生的。

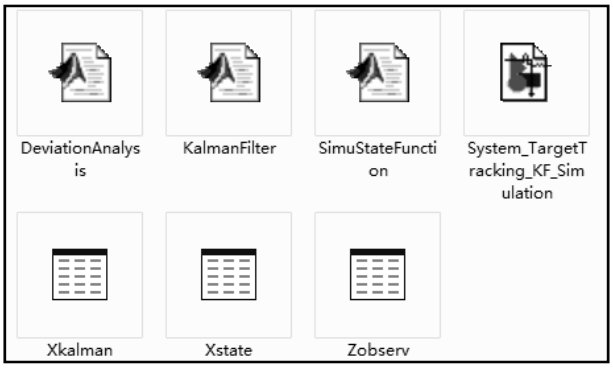


图 7.36　本例所有文件

7.4　非线性 Kalman 滤波

7.4.1　基于 Simulink 的 EKF 滤波器设计

假定一个通用的任意非线性系统，它的状态包含二维变量，即 $\boldsymbol{X}(k)=\left[x_1(k)\quad x_2(k)\right]^{\mathrm{T}}$，状态方程如下。

$$\begin{cases} x_1(k)=x_1(k-1)+1+w_1(k) \\ x_2(k)=x_2(k-1)+\sin(0.1*x_1(k-1))+w_2(k) \end{cases} \tag{7.3}$$

观测方程为

$$Z(k)=\sqrt{x_1^2(k)+x_2^2(k)}+v(k) \tag{7.4}$$

式中，过程噪声 $w_1(k)$ 和 $w_2(k)$ 的均值都为 0，方差分别为 $Q_1=0.01$ 和 $Q_2=0.04$。测量噪声 $v(k)$ 的均值为 0，方差为 R=1。初始状态为 $\boldsymbol{X}(0)=\begin{bmatrix}0 & 0\end{bmatrix}^{\mathrm{T}}$。

式（7.3）和式（7.4）构成的系统是典型的状态方程和观测方程都为非线性的系统。

在 Simulink 环境下构建仿真模型如图 7.37 所示。

其中需要用到两个 Random Number 模块作为过程噪声 $w_1(k)$ 和 $w_2(k)$，1 个 Random Number 模块作为观测噪声 $v(k)$，两个 XY Graph 模块分别显示目标真实状态和 EKF 滤波后的状态。3 个自定义 S 函数模块的程序分别如下。

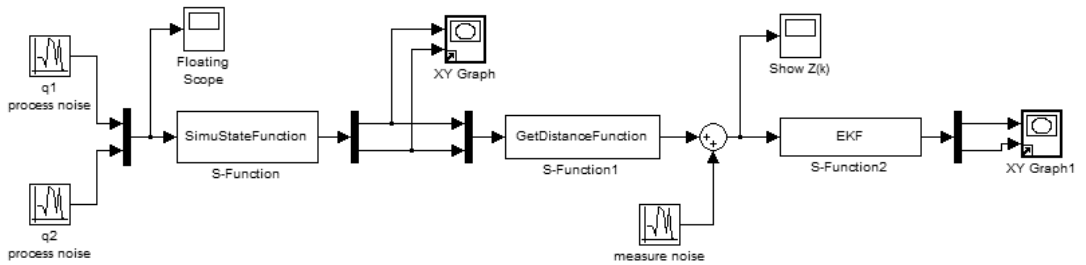


图 7.37　EKF 仿真模型

（1）SimuStateFunction.m

```
% 功能说明：S 函数仿真系统的状态方程
function [sys,x0,str,ts]=SimuStateFunction(t,x,u,flag)
global Xstate;
switch flag
    case 0   % 系统进行初始化，调用 mdlInitializeSizes 函数
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 2   % 更新离散状态变量，调用 mdlUpdate 函数
        sys=mdlUpdate(t,x,u);
    case 3   % 计算 S 函数的输出，调用 mdlOutputs
        sys=mdlOutputs(t,x,u);
    case {1,4}
        sys=[];
    case 9    % 仿真结束，保存状态值
```

```
            save('Xstate','Xstate');
        otherwise     % 其他未知情况处理，用户可以自定义
            error(['Unhandled flag = ',num2str(flag)]);
end
%%%
```

（2）GetDistanceFunction.m

```
% 功能说明：S 函数计算输入信号，并输出距离信息
function [sys,x0,str,ts]=GetDistanceFunction(t,x,u,flag)
switch flag
    case 0   % 系统进行初始化，调用 mdlInitializeSizes 函数
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 2   % 更新离散状态变量，调用 mdlUpdate 函数
        sys=mdlUpdate(t,x,u);
    case 3   % 计算 S 函数的输出，调用 mdlOutputs
        sys=mdlOutputs(t,x,u);
    case {1,4,9}
        sys=[];
    otherwise    % 其他未知情况处理，用户可以自定义
    error(['Unhandled flag = ',num2str(flag)]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1．系统初始化子函数
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 0;    % 无连续量
sizes.NumDiscStates   = 1;    % 离散状态 4 维
sizes.NumOutputs      = 1;    % 输出 4 维，应为状态量是 x-y 方向的位置和速度
sizes.NumInputs       = 2;    % 输入维数，因为噪声模型是 2 维的
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;    % 至少需要的采样时间
sys = simsizes(sizes);
x0   = [0]';                  % 初始条件
str = [];                     % str 总是设置为空
ts   = [-1 0];   % 表示该模块采样时间继承其前的模块采样时间设置
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2．进行离散状态变量的更新
function sys=mdlUpdate(t,x,u)
sys=hfun(u);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3．求取系统的输出信号
```

```
function sys=mdlOutputs(t,x,u)
sys = x;   % 把算得的模块输出向量赋给 sys
%%%
```

```
sizes.NumOutputs      = 2;    % 输出 4 维，应为状态量是 x-y 方向的位置和速度
sizes.NumInputs       = 1;    % 输入维数，因为噪声模型是 2 维的
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;    % 至少需要的采样时间
sys = simsizes(sizes);
x0   = [0,0]';                  % 初始条件
str = [];                       % str 总是设置为空
ts   = [-1 0];   % 表示该模块采样时间继承其前的模块采样时间设置
global Zdist;   % 观测信息
Zdist=[];
global Xekf;      % 粒子滤波估计状态
Xekf=[x0];
global P;
P=zeros(2,2);   % 初始化
%%%
```

```
% 第六步：状态更新
Xnew=Xpre+K*(u-Zpre);
% 第七步：协方差更新
P=(eye(2)-K*H)*Ppre;
% 保存最新的状态并输出
Xekf=[Xekf,Xnew];
sys=Xnew;  % 返回给输出
%%%
```

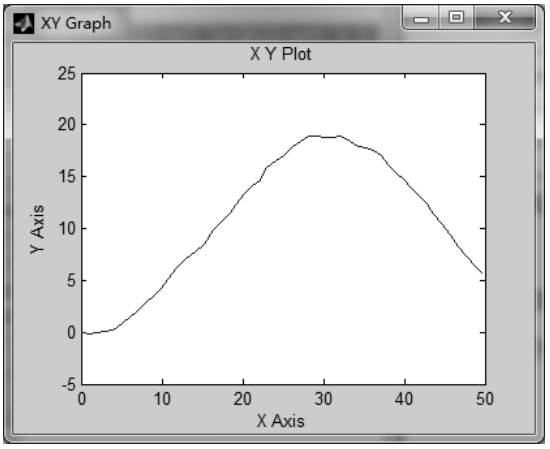


图 7.39　真实状态

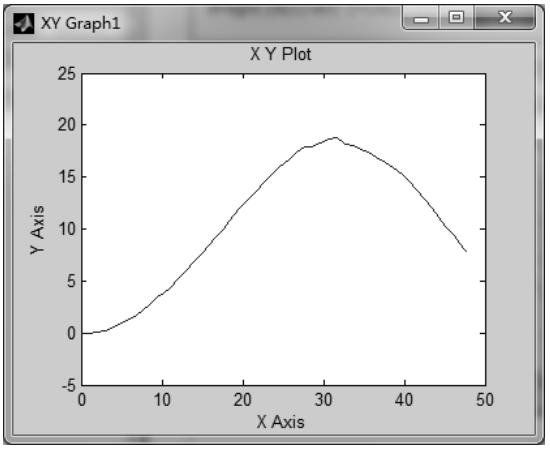


图 7.40　EKF 估计的状态

7.4.2　基于 Simulink 的 UKF 滤波器设计

同样使用非线性系统式(7.3)和(7.4)，构建系统的 Simulink 仿真模型如图 7.41 所示。

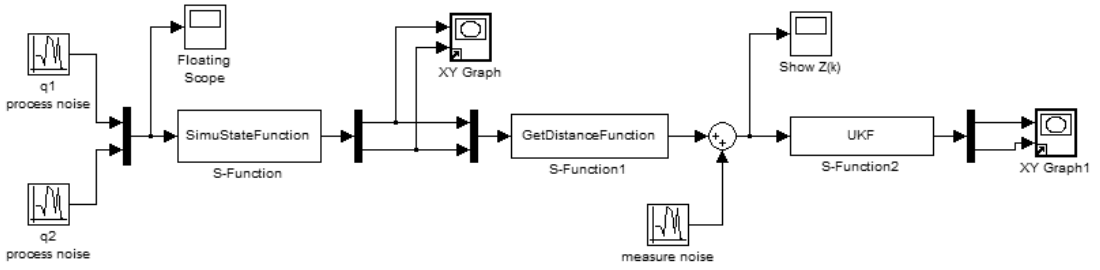


图 7.41　基于 UKF 的非线性系统仿真

用到的 Simulink 模块与 7.4.1 节相似，区别在于自定义函数模块 S-Function2 的 M 文件，即 UKF.m 文件内容如下。

```
%%%
```

```
global Zdist;   % 观测信息
Zdist=[];
global Xukf;      % 粒子滤波估计状态
Xukf=[x0];
global P;
P=0.01*eye(2);   % 初始化
%%%
```

```
% 滤波初始化
xestimate=Xin;
% 第一步：获得一组 sigma 点集
P
cho=(chol(P*(L+ramda)))';
for k=1:L
    xgamaP1(:,k)=xestimate+cho(:,k);
    xgamaP2(:,k)=xestimate-cho(:,k);
end
Xsigma=[xestimate,xgamaP1,xgamaP2]; %Sigma 点集
% 第二步：对 Sigma 点集进行一步预测
for k=1:2*L+1
    Xsigmapre(:,k)=ffun(Xsigma(:,k));
end
%第三步：利用第二步的结果计算均值和协方差
Xpred=zeros(2,1);      % 均值
for k=1:2*L+1
    Xpred=Xpred+Wm(k)*Xsigmapre(:,k);
end
Ppred=zeros(2,2);      % 协方差阵预测
for k=1:2*L+1
    Ppred=Ppred+Wc(k)*(Xsigmapre(:,k)-Xpred)*(Xsigmapre(:,k)-Xpred)';
end
Ppred=Ppred+Q;
% 第四步：根据预测值，再一次使用 UT 变换，得到新的 sigma 点集
chor=(chol((L+ramda)*Ppred))';
for k=1:L
    XaugsigmaP1(:,k)=Xpred+chor(:,k);
    XaugsigmaP2(:,k)=Xpred-chor(:,k);
end
Xaugsigma=[Xpred XaugsigmaP1 XaugsigmaP2];
% 第五步：观测预测
for k=1:2*L+1      %  观测预测
    Zsigmapre(1,k)=hfun(Xaugsigma(:,k));
end
% 第六步：计算观测预测均值和协方差
Zpred=0;           %  观测预测的均值
for k=1:2*L+1
    Zpred=Zpred+Wm(k)*Zsigmapre(1,k);
```

```
end
Pzz=0;
for k=1:2*L+1
    Pzz=Pzz+Wc(k)*(Zsigmapre(1,k)-Zpred)*(Zsigmapre(1,k)-Zpred)';
end
Pzz=Pzz+R;   % 得到协方差 Pzz
Pxz=zeros(2,1);
for k=1:2*L+1
    Pxz=Pxz+Wc(k)*(Xaugsigma(:,k)-Xpred)*(Zsigmapre(1,k)-Zpred)';
end
% 第七步：计算 Kalman 增益
K=Pxz*inv(Pzz);                    % Kalman 增益
%第八步：状态和方差更新
Xout=Xpred+K*(Z-Zpred);            % 状态更新
P=Ppred-K*Pzz*K';                  % 方差更新
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

同样本例的所有程序文件如图 7.42 所示。

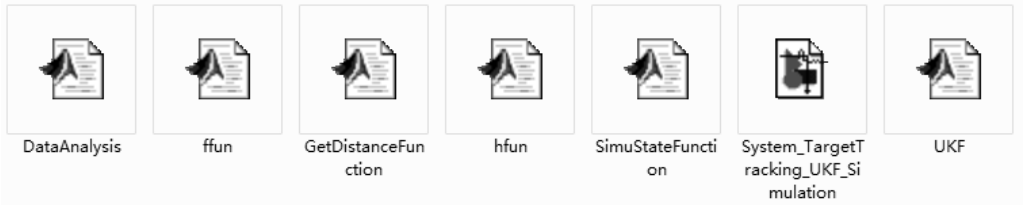


图 7.42　UKF 的非线性系统所有文件列表

运行上面的仿真模型，得到目标的真实状态如图 7.43 所示，UKF 滤波结果如图 7.44 所示。

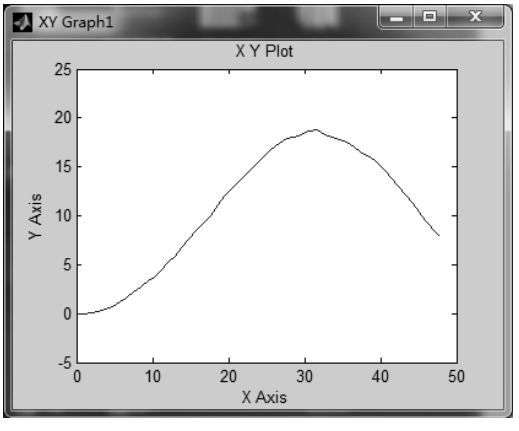


图 7.43　UKF 估计的状态

7.5　小结

本章首先介绍了 Simulink 的基本概念和使用方法，包括系统启动，仿真参数的设置等，同时介绍了基本模块库的使用。接着，重点介绍了 S 函数的工作原理和控制过程，可以说 Simulink 结合自定义模块，几乎能满足所有的系统仿真需求。本章重点是介绍在 Simulink 环境下的 Kalman 滤波仿真，7.3 节给出了线性 Kalman 滤波器的设计方法，7.4 节给出了对于一般特性的非线性系统的 EKF 和 UKF 滤波器设计方法。

反侵权盗版声明